Construction d'un bac à sable à réalité augmentée pour illustrer certains phénomènes hydrologiques, hydrauliques et hydrogéologiques

Auteur : Vivegnis, Elisabeth
Promoteur(s) : Archambeau, Pierre; Pirotton, Michel
Faculté : Faculté des Sciences appliquées
Diplôme : Master en ingénieur civil des constructions, à finalité spécialisée en "civil engineering"
Année académique : 2016-2017
URI/URL : http://hdl.handle.net/2268.2/2629

Université de Liège

Faculté des Sciences Appliquées

Année académique 2016-2017

Construction d'un bac à sable à réalité augmentée pour illustrer certains phénomènes hydrologiques, hydrauliques et hydrogéologiques

Travail de fin d'études réalisé en vue de l'obtention du grade d'Ingénieur Civil des constructions par Elisabeth VIVEGNIS

Composition du jury:

- Pierre ARCHAMBEAU (Université de Liège)
- Michel PIROTTON (Université de Liège)
- Sébastien ERPICUM (Université de Liège)
- Vincent KELNER (Helmo Gramme)

LIÈGE université

Remerciements

La réalisation de ce travail de fin d'études n'aurait pu aboutir sans l'aide apportée. C'est pourquoi je souhaite remercier toutes les personnes qui m'ont aidées durant ces quelques mois.

Tout d'abord, je remercie mon promoteur Monsieur Archambeau pour son suivi attentif, nos échanges et pour ses conseils avisés.

Ensuite, je remercie l'équipe de la Cité des Science de Paris pour leur disponibilité et les échanges que nous avons eus à propos du bac à sable à réalité augmentée dont ils disposent.

Pour la construction du bac à sable, je voudrais remercier mon père pour son aide ainsi que la mise à disposition de ses outils et de son atelier. Le montage du dispositif dans les locaux de l'université n'aurait pas été possible sans l'aide indispensable de mes amies lors de la mise en place et des réglages du système.

Enfin, je souhaite remercier ma famille et mes amis pour leur attitude de compréhension et de soutien durant cette période et plus particulièrement mon père pour son aide précieuse et ses conseils judicieux lors de la relecture de ce travail.

Énoncé

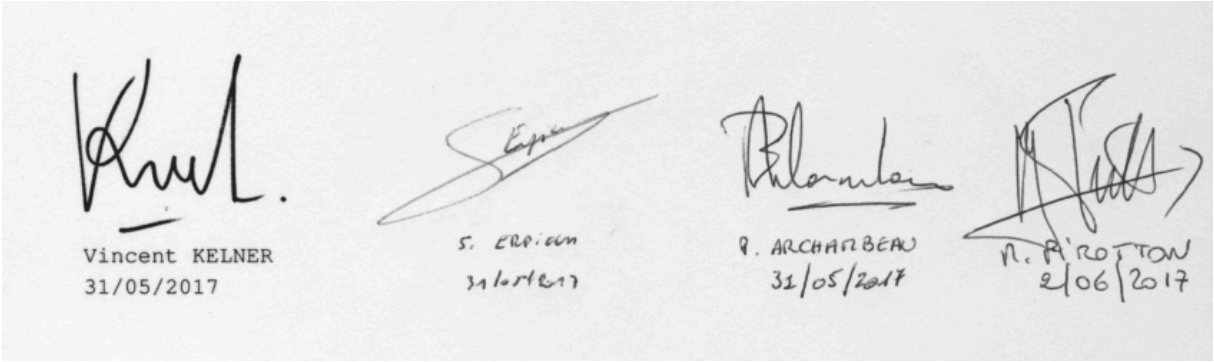
Titre : Construction d'un bac à sable à réalité augmentée pour illustrer certains phénomènes hydrologiques, hydrauliques et hydrogéologiques.

Un bac à sable à réalité augmentée est un outil didactique utilisé pour représenter visuellement le comportement de l'eau sur une topographie modelée. Il permet de montrer les résultats d'une simulation numérique en temps réel de phénomènes hydrologiques et d'études environnementales. Ce dispositif est basé sur la captation d'un relief en sable à l'aide d'une caméra 3D Kinect, d'un traitement numérique par ordinateur en temps réel pour le calcul d'écoulement à partir d'un scénario choisi et la projection du rendu sur le sable par un projecteur vidéo.

Ce travail de fin d'études a deux objectifs principaux.

Le premier concernera la réalisation d'un bac fonctionnel. Pour cette partie, une étude du système existant sera réalisée dans le détail tant au niveau du matériel utilisé que de la structure du programme informatique SARndbox. Ces analyses permettront dans un premier temps de définir le système et les exigences concernant les composants du dispositif afin d'établir les plans des éléments de support à construire. Dans un second temps, les analyses du programme permettront de proposer une adaptation du système existant en vue de l'utilisation d'une version plus récente de la caméra Kinect. A partir de ces résultats, la mise en place d'un bac fonctionnel sera alors effectuée avec une compréhension des paramètres d'exécution.

Le second axe de ce travail étudiera la simulation d'écoulement générée sur le modèle. En commençant par une étude du schéma de résolution des équations d'écoulement, l'étude se poursuivra par la compréhension de l'implémentation du schéma dans le programme informatique. Ensuite, une analyse critique du schéma implémenté sera effectuée. Enfin, une comparaison des résultats du programme à ceux du programme d'écoulement Wolf2D sera établie en imposant les mêmes cas théoriques pour établir les performances du programme du bac à sable à réalité augmentée. Cette dernière étape permettra alors de voir quelles autres utilisations et perspectives sont possibles à partir du système actuel.



Résumé

Le bac à sable à réalité augmentée est un outil à la fois ludique et technologique qui permet de représenter par projection virtuelle 3D couleur, le comportement de l'eau sur une topographie modelée dans le sable et mise à jour en temps réel en représentant des phénomènes hydrologiques, comme la rupture de barrage, les crues, la pluie, en fonction d'interactions avec l'utilisateur.

Dans une première partie du travail, les caractéristiques de la caméra, du projecteur, de l'ordinateur et des pièces mécaniques ont été définies. Un capteur Kinect de deuxième génération en théorie supporté dans le logiciel Open Source SARnbox devait être utilisé mais ne fonctionnait pas. Diverses modifications ont été tentées dans ce logiciel sans succès, seule l'utilisation d'un autre programme utilisant la librairie externe Libfreenect2 a fonctionné pour interagir avec la Kinect. Le bac à sable a donc été construit et est fonctionnel avec une Kinect de première génération.

Dans la deuxième partie du travail, le programme a été modifié pour utiliser une topographie en entrée, exécuter le programme d'écoulement avec des GLS Shader sur GPU et sauvegarder les résultats numériques. En utilisant des topographies de référence, les performances et résultats ont été comparés au logiciel Wolf2D. Des erreurs dans les calculs d'écoulement du programme SARndbox ont été détectées, certains corrigées. Des divergences de résultats entre les deux programmes sont constatées bien que l'allure des résultats est similaire, la comparaison des résultats doit être poursuivie. Au niveau des performances la solution SARndbox sur carte GPU semble beaucoup plus efficace en rapidité de calcul.

Cette première approche d'un bac à sable à réalité augmentée laisse encore de nombreuses opportunités d'investigations et d'améliorations d'un point de vue ludique et universitaire.

The augmented reality sandbox is an educational tool which projects a 3D virtual color image representing the water flow on a topography model realized with sand and updated in real time. It represents hydraulic phenomena like break dam and flood, rain taking into account user's interactions.

In the first part of this master thesis, characteristics of the camera, of the projector, of the computer and of the mechanical pieces were defined. A second generation Kinect 3D camera normally supported in the open source software SARnbox should have been used but was not working. Several modifications were implemented in the software without any success, only the use of another software using an extern library Libfreenect2 worked out to interact with the Kinect V2. The sandbox was built and is functional with the first generation of the 3D Kinect camera.

In the second part of this master thesis, the software was modified to enable to enter a topography file, execute water flow software with GLS Shader on GPU and output numerical results. Using topographies of references, performances and calculation results were compared to the Wolf2D software. Calculation errors in the flow software SARndbox were found, some of which were corrected. Divergences of results between the two software were observed, despite similar shape of results. Comparison must be continued. Regarding the performances, the SARnbox solution on the GPU card seems to be far more efficient in calculation time.

This first approach of the augmented reality sandbox leaves a lot of opportunities of investigation and improvements on an educational as well as university level.

Table des matières

Introduction

Ces dernières années, de nouvelles méthodes d'enseignement pour les jeunes ont été développées pour accrocher la génération actuelle branchée à la technologie. Ces méthodes d'enseignement ludiques permettent alors de capter l'attention et de favoriser l'interaction des élèves. Parmi ces techniques développées, le bac à sable à réalité augmentée est un des outils qui permet d'illustrer visuellement certains phénomènes hydrologiques, hydrauliques et hydrogéologiques comme le phénomène de crue et de sècheresse, les effets d'une rupture de barrage. Un tel outil destiné à des enfants est-il transposable et utilisable dans un contexte universitaire ? La mise en place d'un tel outil didactique se fait-il au détriment d'une reproduction conforme à la réalité et de la précision du modèle pour s'attacher à l'interactivité, l'aspect visuel et ludique ?

C'est dans cette optique que s'inscrit ce travail de fin d'études, il sera développé selon deux objectifs principaux : la mise en place d'un bac à sable à réalité augmentée fonctionnel et l'étude de la simulation d'écoulement proposée.

Le chapitre 1, Etat de l'art, présentera la réalité augmentée ainsi que la caméra Kinect 3D. Une comparaison des deux versions de cette caméra sera proposée suivie de la présentation d'une application dans le domaine de la captation de bathymétrie.

Le chapitre 2 détaille pour chacun des composants du bac à sable, leurs caractéristiques afin d'établir les besoins matériels et les plans des éléments du support à construire.

Le chapitre 3 présente la structure du programme informatique SARndbox qui sera modifié, en détaillant les équations d'écoulement et leur mise en œuvre sur carte graphique.

Une adaptation du système existant en vue de l'utilisation d'une version plus récente de la caméra Kinect sera détaillée au chapitre 4.

Le chapitre 5 présente les différentes étapes aboutissant à la mise en place d'un bac à sable fonctionnel, les fonctionnalités principales sont aussi illustrées.

Le chapitre 6 détaille les modifications apportées au programme en vue de faire une comparaison à ceux du programme d'écoulement Wolf2D. Ce chapitre se clôture par la comparaison de mêmes cas théoriques et un comparatif de performances du programme du bac à sable à réalité augmentée.

La conclusion présentera la synthèse de ce travail et les perspectives d'évolution et d'utilisations futures.

Chapitre 1 État de l'art

Ce chapitre présente la réalité augmentée selon les types de systèmes ainsi que les principales utilisations. Il compare les deux types de caméra 3D Kinect utilisées dans un bac à sable à réalité augmentée et décrit une utilisation d'une Kinect dans le cadre de mesures de surfaces immergées afin de montrer une utilisation possible dans le domaine hydraulique.

1. La réalité augmentée

La réalité augmentée permet de projeter une image virtuelle en 2 dimensions sur un monde réel (dans notre cas un bac avec du sable représentant une portion de topographie). Elle repose sur un système informatique qui permet de superposer un objet virtuel au monde réel tout en intégrant une interaction de l'utilisateur en temps réel avec le système. Un dispositif de réalité augmentée nécessite au préalable une calibration des outils en 3 dimensions pour s'assurer de la superposition parfaite entre les 2 univers. Ce principe de visualisation intègre de nombreuses technologies différentes et s'est développé principalement au cours de ces 10 dernières années avec l'évolution de la technologie et des moyens de calcul. [1]

Il se distingue de la réalité virtuelle qui elle, ne permet que la visualisation en 3 dimensions d'un environnement artificiel et une interaction de l'utilisateur avec ce monde virtuel mais sans aucune superposition au monde réel.

Un système de réalité augmentée est toujours composé des 3 éléments suivants :

- Un système d'acquisition du monde réel composé d'un outil de « tracking » qui permet un alignement parfait entre les objets des 2 mondes nécessaire pour créer une illusion
- Un générateur de rendu sous forme de programme informatique
- L'affichage de la combinaison monde réel/monde virtuel.

La classification des systèmes de réalité augmentée est basée sur la technologie d'affichage, actuellement, on distingue 5 types : [1]

- « Optical See-through » : l'image du monde virtuel est projetée sur des verres de lunette transparents, elle se superpose naturellement au monde réel. Ce système est présenté schématiquement à la Figure 1.

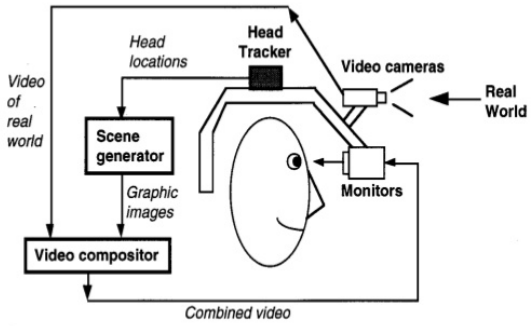


Figure 1: Dispositif optical see-through pour un système de réalité augmentée [1]

- « Virtual-retinal » : l'image virtuelle superposée au monde réel est directement envoyée sur la rétine de l'œil via un jet de lumière modulable. La superposition des deux univers se fait en amont de la projection. L'utilisateur ne voit pas directement le monde réel mais la combinaison des deux mondes. Ce dispositif est présenté à la Figure 2.

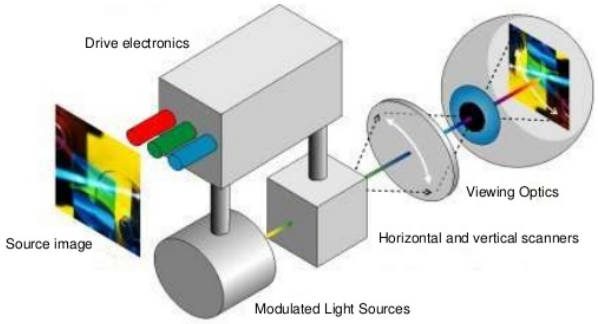


Figure 2: Système virtual-retinal pour un système de réalité augmentée [2]

- « Video See-through » : une caméra capte le monde réel, le système de réalité augmentée réalise la combinaison des 2 mondes. Cette superposition est alors projetée sur des lunettes opaques. Ce dispositif est représenté à la Figure 3.

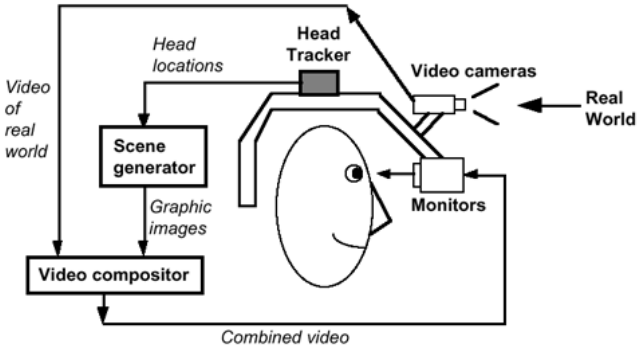


Figure 3: Système video see-through pour un système de réalité augmentée [1]

- « Monitor-based : une caméra capte un espace réel et le système de rendu fusionne cette image et le monde virtuel qu'il a créé. Ce résultat est alors affiché sur une surface plane soit sur un moniteur soit sur un écran avec un projecteur. Ce dispositif est représenté à la Figure 4.

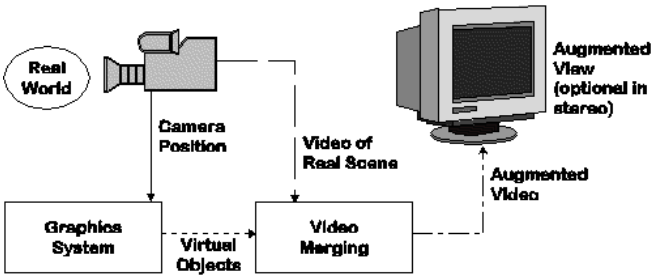


Figure 4: Système monitor-based pour un système de réalité augmentée [1]

- « Projector display » » : il est basé sur le même principe que le système précédent à la différence qu'il utilise les objets réels comme surface de projection pour l'environnement virtuel. C'est dans cette dernière catégorie qu'on peut classer le bac à sable à réalité augmentée.

Avec l'avancée technologique de ces dernières années, la réalité virtuelle a subi un développement important et est utilisée à ce jour dans de nombreux domaines. Ne seront mentionnées dans ce mémoire que les applications les plus courantes.

Un des domaines le plus important de l'utilisation de la réalité augmentée est celui de la médecine, en particulier comme outil de guidage lors d'opérations chirurgicales, par exemple, lors de la localisation d'une tumeur. Elle est aussi utilisée pour des échographies et examens ultrasons, elle permet la visualisation 3D d'un fœtus en mouvement à la position réelle dans le ventre de la mère à l'aide d'un dispositif Optical see-through. La Figure 5 présente la projection des organes internes d'une cheville grâce à un dispositif de réalité augmentée qui utilise différentes radiographies pour créer cette illusion 3D.

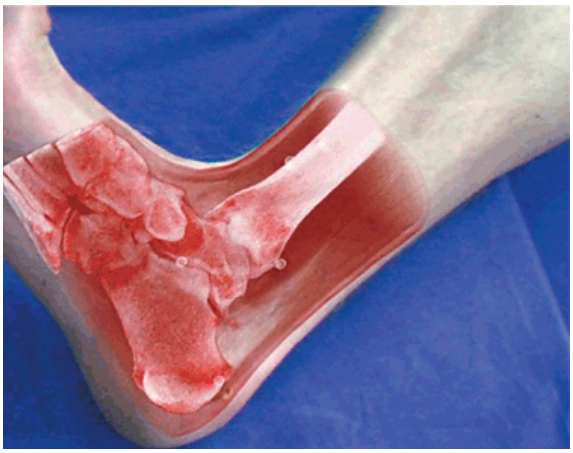


Figure 5 : Visualisation interne d'une cheville grâce à un dispositif de réalité augmentée [3]

Largement utilisée aussi dans le domaine du divertissement, la réalité augmentée est de plus en plus présente dans les jeux vidéo. La firme Microsoft a développé le premier casque à réalité augmentée HoloLens en 2015 et les entreprises de gaming commencent petit à petit à développer des jeux adaptés. Un exemple de rendu d'un jeu vidéo est présenté à la Figure 6.

Figure 6: Visualisation du jeu Young Conker qui utilise la réalité augmentée comme interface de jeu [4]

Ce jeu *(Young Conker)* a été développé par la firme bordelaise ASABO. Le personnage principal évolue sur les meubles et obstacles présents dans la pièce. Ils sont détectés grâce à la caméra présente sur casque. [4]

Citons aussi d'autres applications :

- Simulateurs de vols avec cockpit virtuel dans le domaine de l'aviation civile ou militaire
- Visualisation du rendu d'une pièce dans le domaine du design quoi que peu utilisé à ce jour
- Guidage de robots ou bras articulés pour des opérations chirurgicales [1]

2. L'utilisation d'un capteur Kinect

Dans les systèmes de réalité augmentée, une caméra ou un capteur est utilisé afin de pouvoir capter le monde réel. Afin de traiter cette image, pour par exemple lui superposer une image virtuelle, il est donc nécessaire que les données de l'image soient exploitables de manière numérique (position dans l'espace). Des caméras 3D ou scanners ont été utilisés par le passé pour obtenir les données du monde réel à savoir la distance d'objet et la position de celui-ci par rapport à un capteur. Aujourd'hui, dans le domaine expérimental, le capteur Kinect 3D est le plus utilisé de par son prix très abordable contrairement au caméras 3D. [5]

Le premier capteur appelé Kinect for Xbox 360, aussi appelé Kinect v1, a été développé par Microsoft en 2010, il est à la fois composé d'un capteur de profondeur et d'une caméra couleur. Son avantage par rapport aux caméras 3D utilisées précédemment est son faible cout, environ 10 fois moins qu'une caméra 3D. D'abord utilisée comme accessoire de jeu vidéo permettant un suivi des mouvements du joueur utilisant une console Xbox, elle a ensuite été utilisée dans d'autres applications. En effet, plus ou moins 1 an après la sortie du la première version, Microsoft a fourni un outil de développement logiciel appelé ''Kinect for Windows'' pour créer des applications utilisant la Kinect. [5]. En 2013, une nouvelle version de la Kinect est mise sur le marché pour remplacer la

première version, appelée Kinect v2 ou Kinect for Xbox One afin d'améliorer la précision des mesures de distance et la résolution de la caméra couleur. Dans cette version, la technologie de captation de profondeurs est totalement différente de la première version mais elle utilise les mêmes outils. En 2014, une nouvelle version de l'outil de développement est sortie.

Le capteur est composé d'une caméra RGB ainsi que d'un émetteur et récepteur infrarouge permettant de capter à la fois des images couleurs et les distances de la surface captée par rapport au capteur.

Pour interagir avec la Kinect à partir d'un ordinateur, il existe une librairie officielle développée par Microsoft uniquement sur operating system Windows. Pour les autres operating system, des librairies multiplateformes, propres à chaque version et open source ont vu le jour au moins 1 an après la sortie de l'outil de développement officiel comme Libfreenect, Libfreenect2, OpenNI, NITE…

L'utilisation de la Kinect permet 2 fonctions principales qui ont fait son succès au cours des années :

- Le tracking des mouvements d'un squelette
- La captation de distance profondeur d'une surface par rapport au capteur. [5]

De nos jours, le capteur Kinect est utilisé dans de nombreuses applications de domaines très différents, citons parmi les plus importantes : [6]

- Le mapping 3D intérieur
- La modélisation 3D
- La santé
- La surveillance
- Les sciences de la Terre
- La biométrie
- …

Le tracking du mouvement d'un squelette est utilisé par exemple dans le jeu vidéo à réalité virtuelle « JewelMine ». Ce jeu est utilisé pour la rééducation de personnes qui ont subi des traumas neurologiques. Grâce à leur mouvement elles arrivent à toucher des cibles virtuelles. La Kinect traque alors les mouvements du squelette du patient à la fois pour le rendu de l'image dans l'application et pour proposer des exercices en fonction de l'analyse des images. [7]

Dans le domaine des sciences de la Terre, le capteur est utilisé dans 3 applications principales : [6] [8]

- Le streaming de bathymétrie et de surface immergée
- La glaciologie
- La géomorphologie.

2.1 Comparaison des capteurs Kinect

Les 2 versions de capteur possèdent quasi les mêmes composantes :

- Une caméra couleur (RGB)
- Un émetteur infrarouge
- Un récepteur infrarouge
- Un microphone
- Un moteur à bascule (Xbox 360 seulement) qui permet le mouvement de haut en bas du capteur par rapport au socle pour adapter la perception de la caméra en fonction de la position de l'objet dans la pièce.
- Un accéléromètre 3D

La Figure 7 présente une image des 2 versions des capteurs.

Figure 7 : Kinect for X Box 360 (à gauche) [9] Kinect for X Box One (à droite) [10]

Le Tableau 1 présente les caractéristiques principales des 2 versions de capteurs. Se référer à l'Annexe 1 pour un détail des caractéristiques de chaque capteur.

Caractéristiques	Kinect for Xbox 360 (V1)	Kinect for Xbox One (V2)
Dimensions (mm)	180x 25 x 35	249 x 66 x 67
Caméra RGB (pixel)	1280 x 1024 ou 640 x 480	1920 x 1080
Caméra Infrarouge (pixel)	640x480	512 x 424
Distance maximale de profondeur (m)	4,8	4,5
Distance minimale de profondeur (m)	0,5	0,5
Champ visuel horizontal (degré)	57	70
Champ visuel vertical	43	60
Vitesse de captation (frame/seconde)	30	Variable entre 15 et 30
Moteur à basculement	Oui	Non
USB	2.0	3.0
Prix (€)	30	199

Tableau 1 : Comparaison de la Kinect V1 et V2

2.1.a Caméra Infrarouge

Fondamentalement, la technologie de captation des profondeurs est très différente entre les 2 versions. Dans la première version, basée sur une méthode de triangulation, un faisceau infrarouge est émis vers la surface sous forme d'une multitude de faisceaux obtenus par diffraction. Ce faisceau crée un motif sur la surface captée (Figure 8 à gauche) qui est alors capté par la caméra et qui le traite grâce à une comparaison avec un motif de référence. Celui-ci est stocké dans la mémoire interne du capteur et a été obtenu grâce la capture d'une surface à une distance connue, cette procédure est effectuée avant commercialisation du produit. La mire projetée sur la surface par la caméra infrarouge et le résultat de l'image des profondeurs sont visibles à Figure 8. [11]

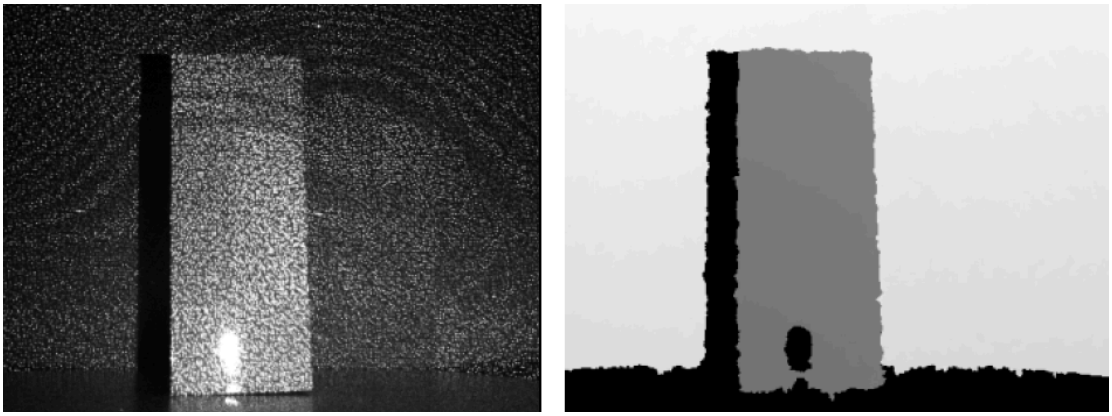


Figure 8: Mire infrarouge émise (à gauche) par un capteur Kinect et image de profondeur résultante (à droite) [11]

Si la distance d'un objet au capteur est plus courte que celle au plan de référence alors l'ensemble du spectre de points sera décalé dans l'image infrarouge dans le sens de la ligne passant la caméra infrarouge et l 'émetteur IR. Ces décalages, appelés disparités (d sur la Figure 9), sont mesurés. Effectués pour chaque point, une image de disparités est obtenue. Ces données de décalage résultent alors d'une expression basée sur le théorème de Thales à partir de la mesure de disparité et des données de référence. Cette procédure est établie pour chaque pixel afin de connaître la distance au capteur. La Figure 9 présente la relation entre disparité et distance au capteur. [11]

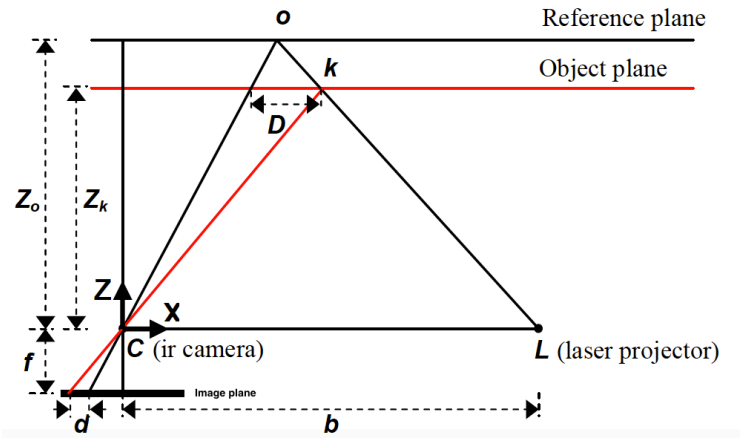


Figure 9: Relation entre la disparité (d) dans l'espace image et la distance (Z_K) au capteur d'un objet placé en k [11]

Cependant, ce type de mesure induit beaucoup des pixels (+/- 1/20 du nombre de pixels) sans valeurs si le faisceau obtenu par diffraction n'est pas robuste assez. Il est nécessaire de traiter donc plusieurs images de profondeur (frame) à la fois pour obtenir une image suffisamment complète et stable. [12]

Dans la seconde version de la Kinect, la technologie est beaucoup plus simple et basée sur le temps de vol. Ce principe de captation est illustré à la Figure 10. Un signal infrarouge est envoyé sur la surface et l'analyse du signal réfléchi permet de déterminer la distance au capteur avec une expression mathématique dépendant de la fréquence d'obtention des images (variable), de la vitesse de la lumière et phase du signal reçu mesurée par le capteur. [12] [10]

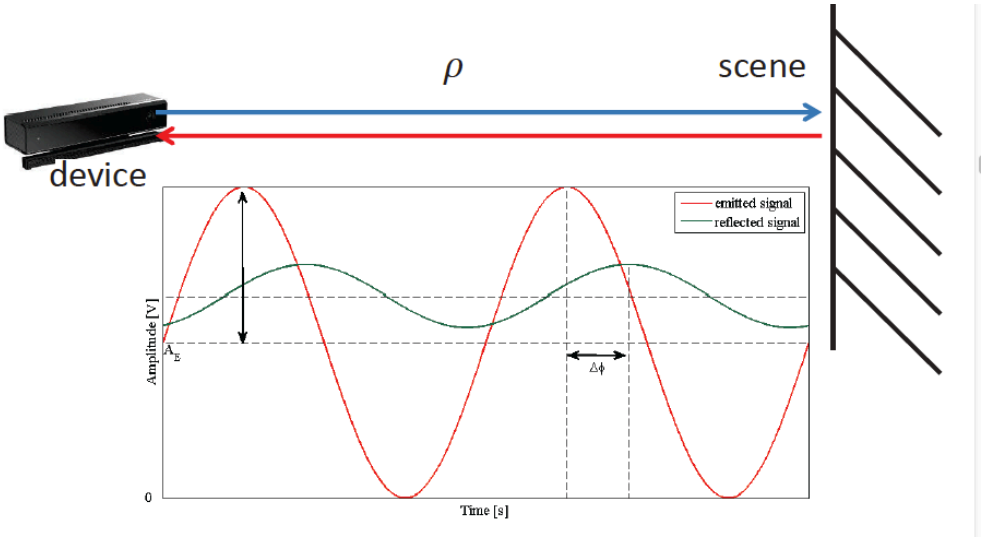


Figure 10: Principe de la caméra temps de vol de la Kinect v2 [10]

Des images de profondeurs pour chacune des versions sont disponibles à la Figure 11 avec en bleu marine les pixels sans données. On peut dire, suite à l'étude de la comparaison de 100 images de profondeurs des 2 versions de capteurs établies [12] que la seconde version possède beaucoup moins de pixels sans données en plus de présenter plus de détails du relief de la surface captée.

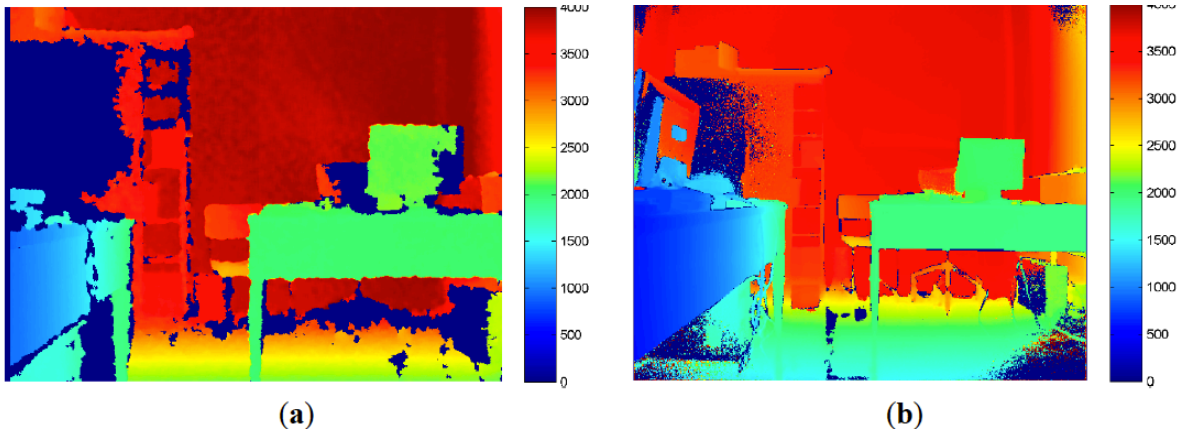


Figure 11: Image de profondeur obtenue par une Kinect a) v1 b) v2 [12]

2.1.b Caméra couleur

La résolution de la caméra couleur a été améliorée en passant d'une résolution 640x480 pixels dans la première version à une résolution de 1920x1080 pixels càd de la haute définition dans la seconde version. Une comparaison des 2 images avec leur capteur située à la même position est présentée à la Figure 12.

Figure 12: Comparaison des images obtenues par la caméra RBG avec une Kinect a) v1 b) v2 [12]

Dans le cas de la seconde version, le champ de vue de la caméra RGB et infrarouge ne se chevauchent pas parfaitement : la caméra infrarouge à un champ plus large verticalement et plus étroit horizontalement comme présenté à la Figure 13. [12] Cette superposition non parfaite existe aussi dans la première version mais de manière moins exagérée.

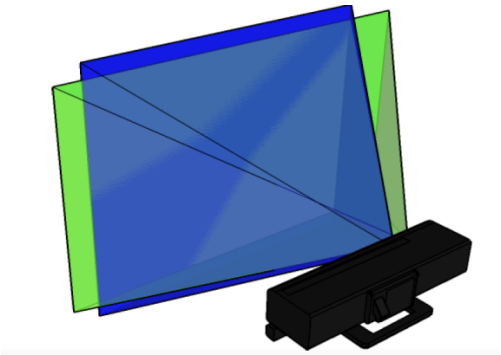


Figure 13: Champ de vue obtenue par un Kinect v2, caméra couleur (en vert) et caméra IR (en bleu) [12]

2.1.c L'éclairage de la surface

Pour la première version, la littérature indique que le motif de lumière infrarouge projetée sur la surface à capter dépend fortement de l'ensoleillement. La Kinect v1 n'est pas efficace en extérieure. Pour la seconde version, on remarque qu'elle détecte plus de détails dans les mêmes conditions d'ensoleillement. Il a aussi été montré qu'en extérieur, s'il y a peu de radiation infrarouge, le capteur fonctionne très bien. Cependant en cas de lumière directe, le capteur fait difficilement la différence de radiations provenant de l'émission du capteur et celle de l'ambiance, et ceci dépend fortement de l'orientation de la lumière. [12]

2.1.d La précision des mesures de profondeurs

Dans le Tableau 1, on remarque que pour obtenir des images exploitables, la distance entre le capteur et la surface captée doit être comprise entre 0,5m et 4,8m pour la première version des capteurs et entre 0,5 et 4,5m pour la seconde. [11] [12]. Une étude [12] a étudié l'erreur de distances mesurées par chaque type de capteur, en analysant les centres de 100 images de profondeurs (200x200 pixels). La Figure 14 présente l'erreur de mesures en fonction de la distance entre la caméra et la surface captée. On remarque une erreur beaucoup plus faible dans la cas d'une Kinect v2 par rapport à la v1. Par interpolation des valeurs, l'erreur semble suivre une loi polynomiale pour la première version et une loi linéaire pour la v2. Ce qui donne par exemple, pour un capteur placé à 2 m de la surface observée, une précision de 1 cm avec la seconde génération de capteur et de 3cm avec la première génération. En dessous de 1m, on obtient des erreurs en dessous du centimètre [12] [10]

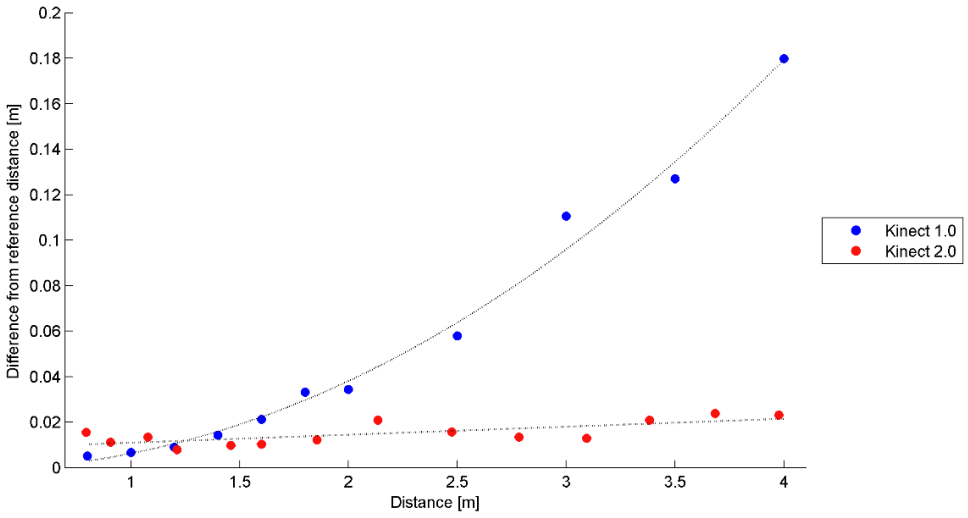


Figure 14: Erreur de mesures de profondeurs en fonction de la distance au capteur selon le type de capteur Kinect [11]

2.1.e Etude des imprécisions des capteurs

Les images de profondeur captées par une Kinect possèdent du bruit d'image constitué d'informations parasites qui se superposent aléatoirement à l'image prise.

Une quantification de ce bruit a été mise en évidence par l'étude de la précision des valeurs d'un pixel par rapport aux autres et ce pour toute l'image. L'étude de 100 images de 200 x 200 pixels, explicitée dans [12], a montré que le bruit d'image issu du capteur Kinect v2 est dix fois inférieur à celui de la première version pour une distance de 3,5m de la surface à la caméra. La Figure 15 présente la variance des mesures de profondeur en fonction de la distance au capteur. Par interpolation des valeurs, dans la première version, cette variance suit une loi polynomiale du second ordre et dans la seconde, elle suit une fonction linéaire. [12] [10]

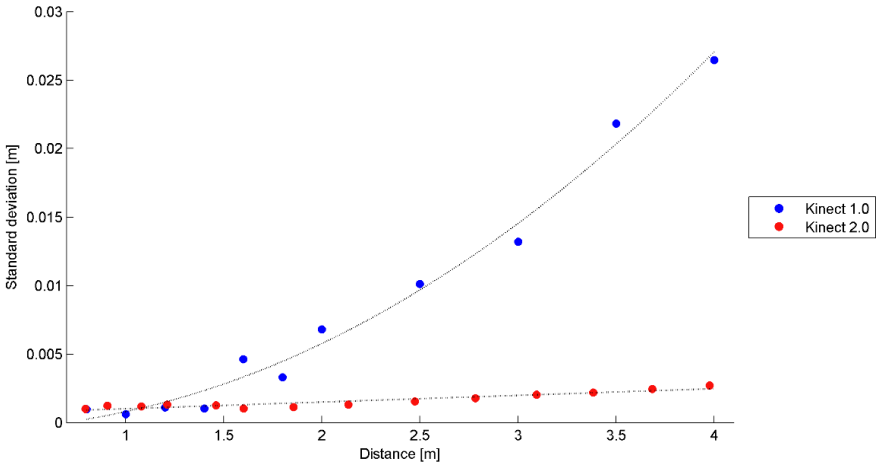


Figure 15: Variance des mesures de profondeurs en fonction de la distance au capteur selon le type de capteur Kinect [11]

L'étude a aussi montré des écarts de la variance dans le champ visuel capté. La Figure 16 présente cette répartition de variance sur la surface captée dans le cas d'une Kinect seconde génération placée à 90 cm de la surface et parallèle au capteur. On remarque bien que le bruit est massivement présent dans les coins par rapport au centre de l'image et qu'il augmente quand la distance à la Kinect augmente. Cependant, les valeurs de variance restent faibles (<5 mm) pour être négligées dans l'utilisation d'une Kinect v2 au sein d'une application.

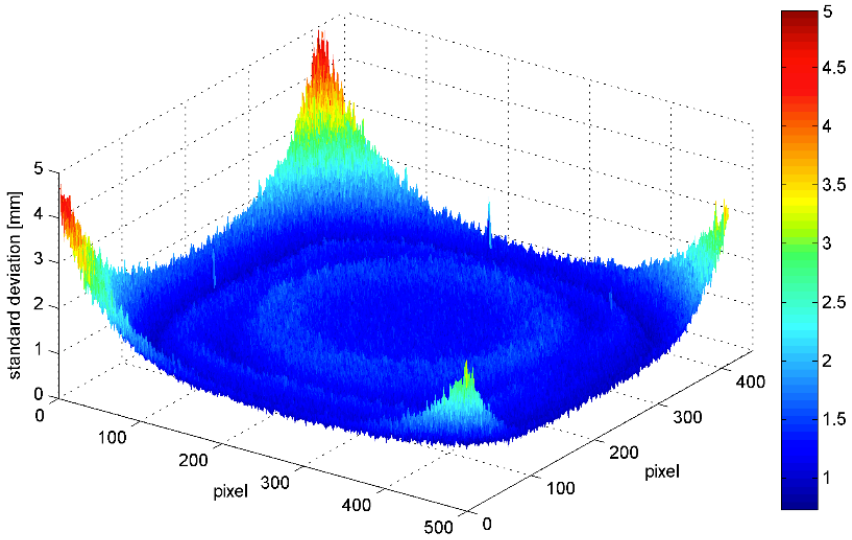


Figure 16: Variance des valeurs de chaque pixel obtenu par une Kinect v2, dans le cas d'une surface parallèle au capteur et situé à une distance de 90 cm. [11]

L'amélioration de la précision des mesures de profondeurs en passant de la première à la seconde génération de Kinect tant au niveau de la distance captée que de la variance des valeurs des pixels est très nette. Cependant, il est à noter que la Kinect seconde génération est encore faiblement utilisée comme les premières librairies Open Source ont seulement été développées début 2016.

2.2 Utilisation de la Kinect pour le streaming de surface immergée

Assez récemment, la Kinect v1 a été utilisée pour étudier l'évolution d'un fond de rivière mobile dû à écoulement d'eau à l'aide d'une Kinect. En effet, l'eau présente au-dessus du fond entraine des erreurs de mesures de profondeur directement rendues par le capteur. Une étude a été réalisée pour interpréter les effets de différents paramètres sur les mesures dans l'air et dans l'eau ainsi que pour proposer des corrections afin de corriger les erreurs de mesures, pour établir une procédure de l'utilisation de la Kinect dans un contexte hydraulique de type « shallow water ».

Lors de cette étude réalisée par des chercheurs indiens, une expérience a été réalisée dans le laboratoire d'hydraulique de l'institut indien de l'ingénierie de Kanpur en Inde. Le dispositif est présenté à la Figure 17. Il est composé d'un canal vitré de 1,5m de long et 30 cm de large avec en son milieu un objet collé au fond du canal, 3 différentes formes ont été testées. Au-dessus de l'objet, la Kinect v1 a été disposée à une hauteur de plus ou moins 60 cm afin d'avoir la meilleure précision. Le niveau d'eau dans le canal a été établi en garantissant un débit entre 2 et 3,5 litres par seconde. [6]

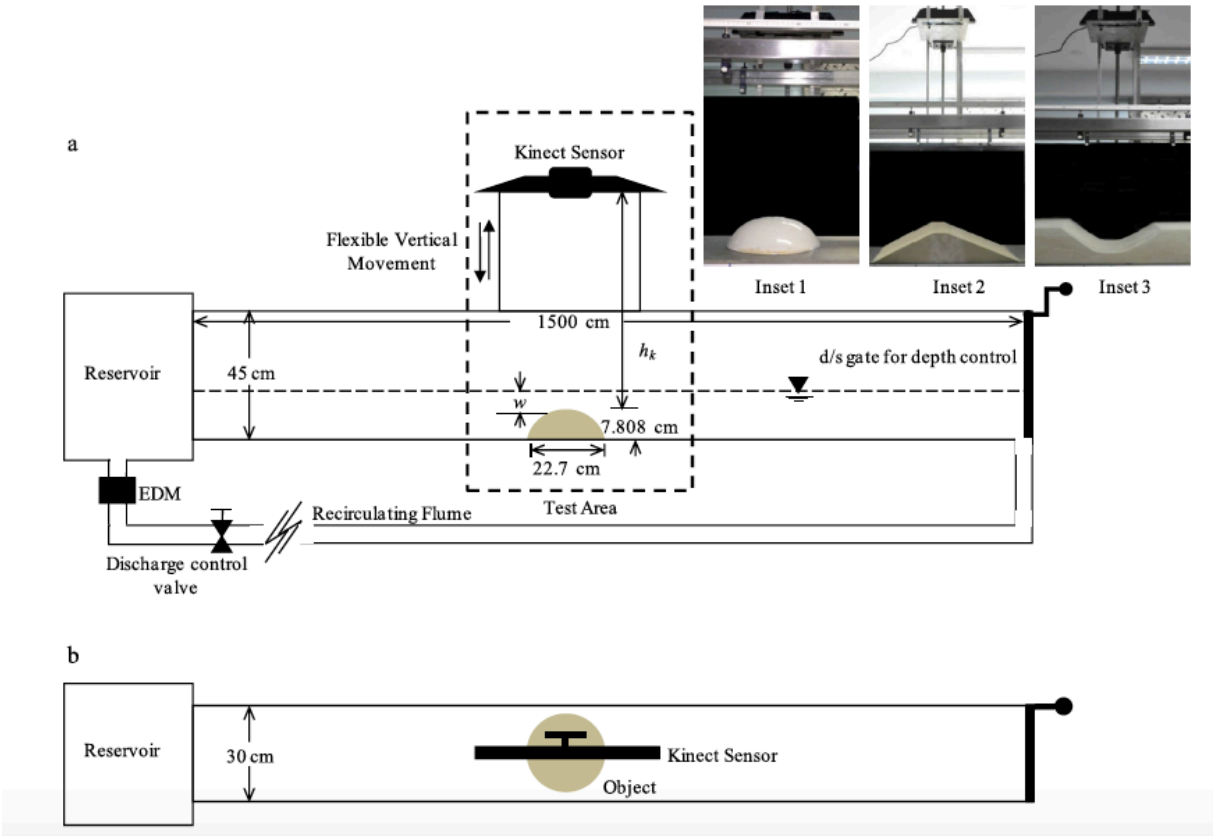


Figure 17: Dispositif expérimental pour la mesure de surface immergée à l'aide d'un capteur Kinect v1. a) vue en élévation b) vue en plan. [6]

Sept expériences différentes ont été menées afin de mettre en évidence des comportements et des influences de certains paramètres. Les objectifs de ces expériences étaient quintuples. [6]

1. Positionner idéalement la Kinect : il a été établi une formule pour déterminer la distance idéale du capteur par rapport au fond du lit en fonction de la surface à couvrir. Pour assurer une précision dans les mesures de +/-2 mm, cette distance devait être comprise entre 50 cm et 175 cm. [6]
2. Réduire les erreurs de mesures pour les mesures dans l'air : il est apparu que la Kinect ne captait pas certains points et la proportion de ces points noirs était inférieure à 5% de la surface couverte comme présenté à la Figure 18. Cette faible erreur peut être diminuée grâce à une méthode « locally weighted scatter-plot smoothing » (loess) qui consiste en une régression linéaire entre les points avec des poids différents alloués selon la distance avec le point d'observation. [6]

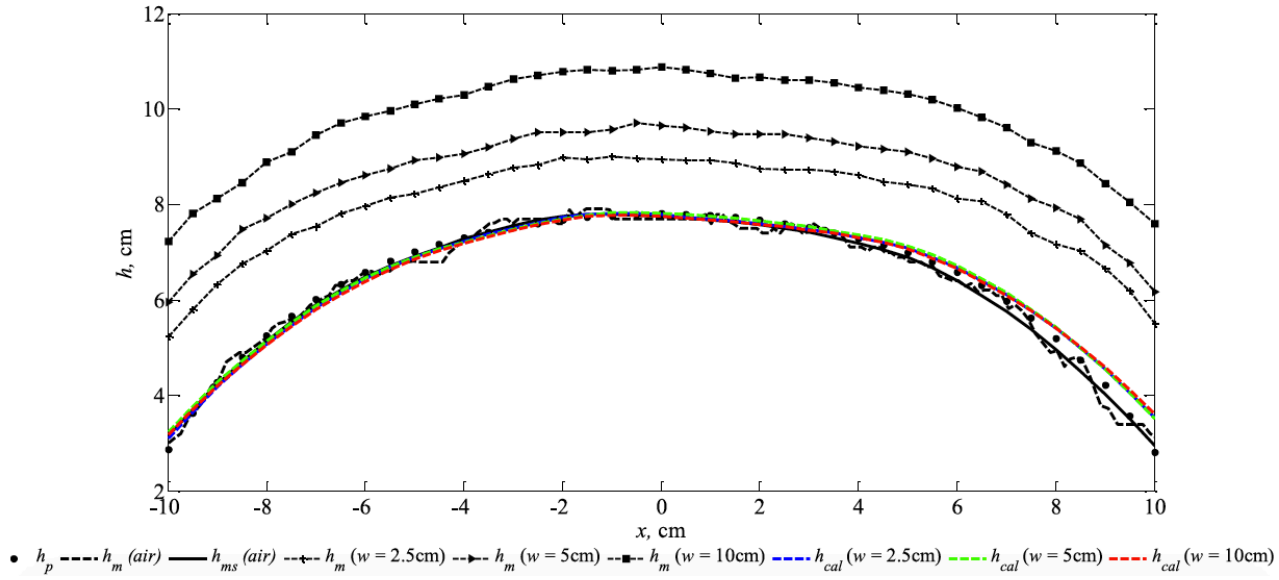


Figure 18: Mesures obtenues par un capteur Kinect d'une expérience de mesures sous eau : h_p mesure d'une gauge, h_m mesure brute du capteur, h_{ms} mesure avec correction grâce la méthode loess, h_{cal} mesure avec correction du la réfraction pour différence profondeur d'eau (w)

3. Développer un algorithme de correction pour la prise en compte de la réfraction dans l'eau et analyser la précision de celui-ci. Il a été observé que même pour une faible profondeur d'eau (2cm) il y a une déviation dans les mesures due au phénomène de réfraction de la lumière à la surface de l'eau à la Figure 18 avec les trois courbes en parties supérieures du graphe. Ces erreurs sont d'autant plus grandes que la profondeur d'eau augmente. Elles sont liées au phénomène de réfraction : le capteur ne capte que la distance apparente (a sur la Figure 19) et non la distance réelle (w sur la Figure 19) comme présenté à la Figure 19 . Sur base d'une étude de la réfraction de la lumière, une expression mathématique a pu être extraite pour déterminer la hauteur corrigée. [6]

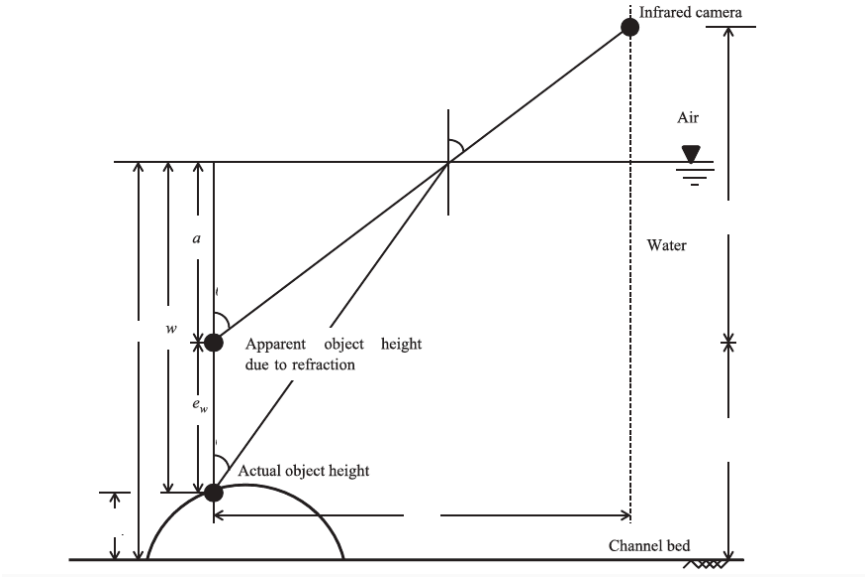


Figure 19 : Modèle de réfraction de lumière avec l'utilisation d'une Kinect [6]

4. Etudier l'influence des paramètres de hauteur du capteur par rapport à l'objet, de couleur de l'objet et d'illumination de la surface. Il a été démontré que la couleur de l'objet n'a aucune influence significative sur les mesures et que la hauteur du capteur comprise entre 50cm et 100cm assure une précision de 2mm des mesures. Concernant l'illumination de la surface, elle influe sur la densité de pixels et celle-ci diminue quand l'illumination augmente. Pour avoir une bonne densité, il faut maintenir l'illumination entre 0 et 750 lux . [6]
5. Examiner l'influence de la turbidité de l'eau : la turbidité diminue le taux de pénétration de l'infrarouge, alors qu'elle augmente avec la profondeur. Une expression a été établie pour déterminer la profondeur maximale pour qu'un objet soit détecté, avec au moins 50% des points de l'image qui ne soient pas des pixels sans valeurs. Pour une turbidité plus faible que 51 NTU (Nephelometric Turbidity Unit), il n'y a pas de limitation de profondeur d'eau et pour une turbidité supérieure à 475 NTU, à cause de l'opacité de l'eau, le capteur Kinect capte la surface de l'eau. [6]

On peut donc dire qu'une utilisation de la Kinect est envisageable pour mesurer les évolutions de bathymétrie de lit de rivière mobile et statique dans des conditions d'eau peu profonde. Dans des conditions strictes, les conditions minimales afin de garantir une certaine précision des mesures sont :

- Turbidité de l'eau inférieure à 50 NTU
- Illumination de la surface inférieure à 750 lux
- Surface captation limitée par la distance de la surface au capteur. Si la distance est égale à 1m alors la surface captée doit être inférieure à 1m x 0,75 m.

Dans le cas de mesure de surface immergée, des équations de corrections des valeurs obtenues doivent être implémentées pour palier à la réfraction de la lumière dans l'eau ainsi qu'un modèle pour améliorer la précision des mesures dans le cas d'utilisation de la Kinect de première génération.

Chapitre 2 Le bac à sable à réalité augmentée

1. Description du concept

Le bac à sable à réalité augmentée appelé Sandbox est un outil didactique utilisé pour représenter de manière numérique le comportement de l'eau sur une topographie modelée dans le sable et mise à jour en temps réel grâce à la présentation de phénomènes hydrologiques. La Figure 20 présente une photo d'un tel dispositif.

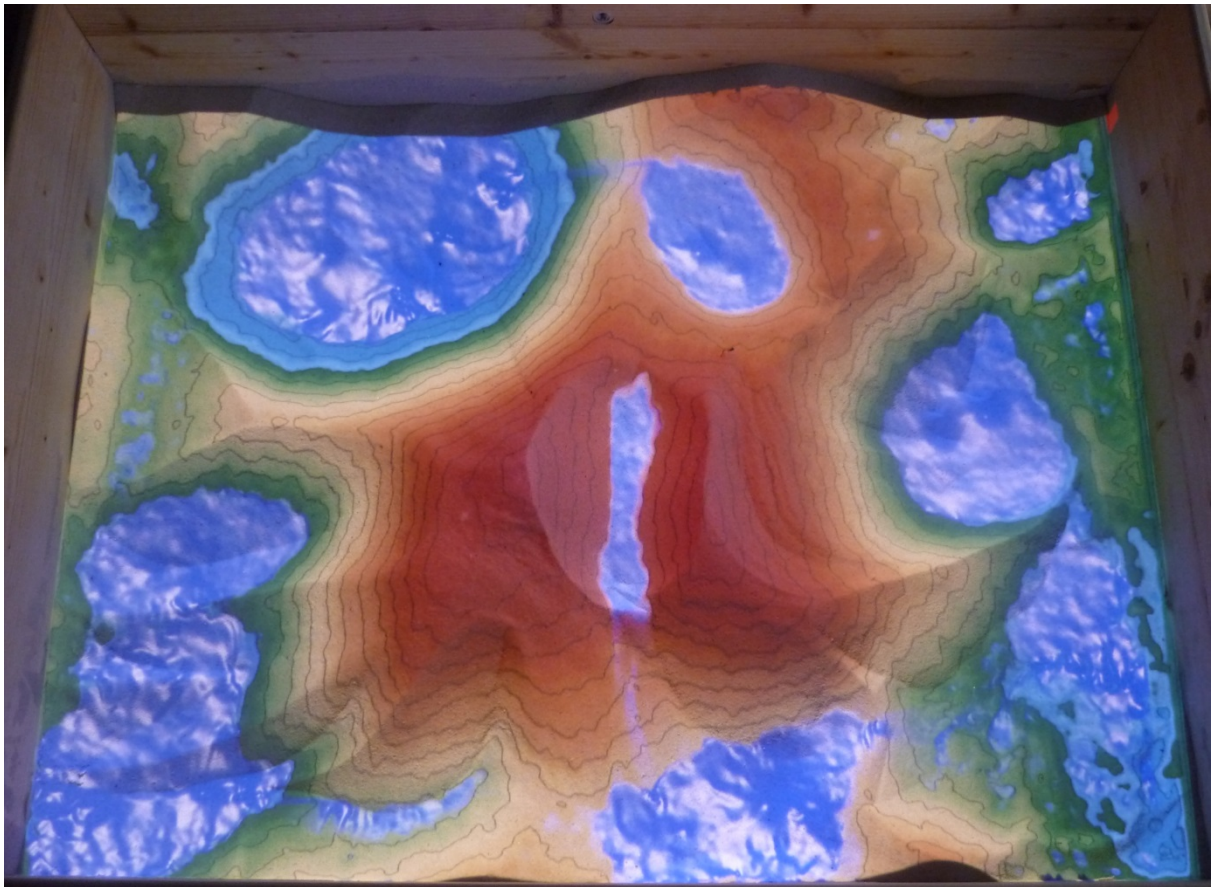


Figure 20: Bac à sable à réalité augmentée

Ce système est un dispositif de réalité de type Projector Display comme présenté au Chapitre 1 -1. La Figure 21 présente les principaux éléments du dispositif.

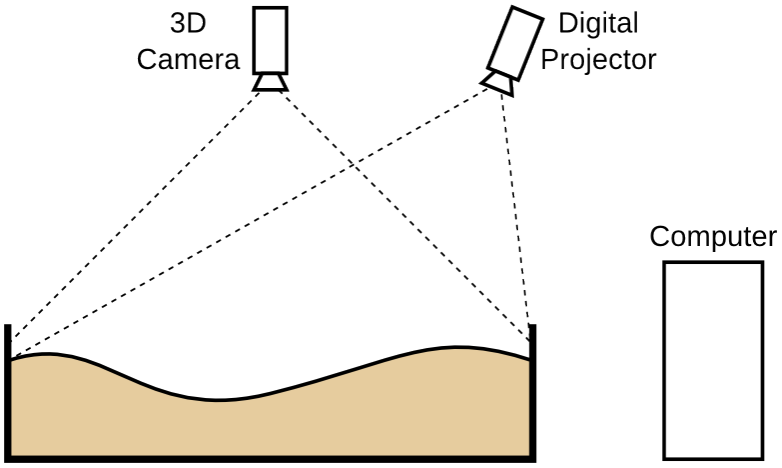


Figure 21: Dispositif d'un bac à sable à réalité augmentée [13]

Le fonctionnement d'un bac est composé d'une suite d'opérations effectuées en temps réel et en boucle dès le lancement de l'application SARndbox installée sur un ordinateur et jusqu'à sa fermeture :

1. Un relief quelconque est modelé ou modifié par l'utilisateur sur le sable présent dans un bac rectangulaire sur une épaisseur de 10 cm.
2. Une caméra Kinect placée au-dessus du bac acquiert des frames avec une vitesse de captations de 30 frames par seconde. Un frame est une image dont chaque pixel contient une information soit une distance d'un point de la surface captée de la caméra soit les données de la couleur captée.
3. Le programme informatique SARndbox extrait les données du frame concernant la distance. Il traite alors ces données en corrigeant les imprécisions de mesures dues au capteur et complète les données des pixels sans valeurs.
4. Le programme informatique réalise alors les deux opérations suivantes en parallèles
    a. Création de rendu de la surface topographique corrigée avec des nuances de couleurs et des courbes de niveaux.
    b. Calcul d'un écoulement d'eau sur la surface (si l'option est utilisée) avec une création de rendu de l'eau avec des textures et reflets de l'eau.
5. Le projecteur projette un rendu coloré de la topographie sur la surface de sable.

Il faut noter qu'avant toute utilisation, une étape de calibration entre la Kinect et le projecteur doit être réalisée.

Une explication plus détaillée de cette procédure du point de vue informatique sera détaillée dans le Chapitre 3 2.1 .

Les principales fonctionnalités d'une Sandbox [1] sont dans un premier temps la possibilité de « faire pleuvoir » en imposant l'ajout d'eau virtuelle sur toute la surface captée en choisssisant une intensité de précipitations. Ensuite, si la main de l'utilisateur, avec les doigts écartés, est placée à plus de 15 cm au-dessus de la surface de sable, la Kinect détecte cette main et l'associe à un nuage pluvieux, un changement de topographie au droit de la main n'est pas intégré dans la topographie de la surface représentée ni dans le calcul d'écoulement. Cependant, un disque de pluie est généré en cet endroit de la surface.

Il est aussi possible d'importer une topographie réelle (Digital Elevation Model, DEM) sous un format raster et de la représenter sur la surface de sable. Cette projection permet alors à l'utilisateur de modifier la forme du sable pour qu'elle corresponde à la topographie projetée. En effet, des zones en bleu correspondent à une hauteur de sable trop importante et en rouge une hauteur trop faible.

Les avantages d'un tel bac sont multiples :

- Le fonctionnement en temps réel permet un rendu quasi instantané en cas de modification de la forme du sable
- L'utilisation d'eau virtuelle rend le modèle réaliste et favorise une compréhension plus rapide et plus pérenne des concepts expliqués.

1.1 Historique

Le concept du bac à sable à réalité a été développé en 2012 par le chercheur Oliver Kreylos de l'université de Californie (UC), Davis aux Etats-Unis. Ce scientifique informaticien est spécialisé dans la visualisation 3D et dans les géosciences informatiques. Des améliorations à la première version ont été menées en collaboration avec l'équipe de LakeViz3D. [13]

Des mises à jour ont été effectuées au fur et à mesure. Le tableau à l'Annexe 2 liste les différentes versions et mises à jour ainsi que les améliorations et changements entre versions.

Actuellement, la dernière version du logiciel est la version 2.3. De nombreuses Sandbox ont été reproduites à travers le monde. La carte présentée à la Figure 22 localise les Sandbox réalisées dans la partie Ouest de l'Europe qui ont été déclarées auprès de la communauté. En Belgique, à ce jour il n'existe qu'actuellement 2 bacs recensés, un dans la région flamande (Aalst) et un à l'université de Liège réalisé lors de ce travail de fin d'étude.

[1] L'utilisation du terme Sandbox dans ce travail se réfère à tout le dispositif ainsi que l'interaction entre les composantes via le programme informatique.

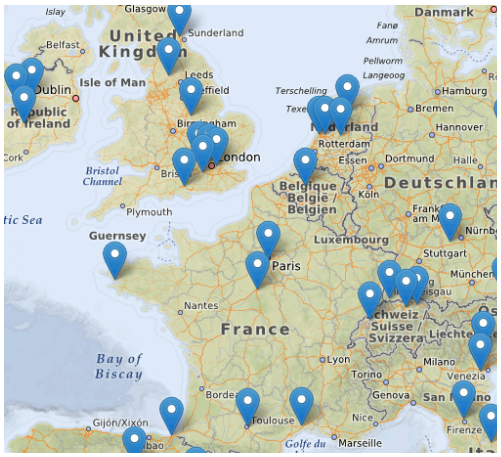


Figure 22: Localisation des bacs à sable à réalité augmentées en Europe de l'Ouest [13]

Les améliorations futures prévues pour ce logiciel sont : [13]

- la prise en compte d'autres types de caméras 3D comme la Kinect seconde génération et les caméras Intel RealSense
- la détection d'intensité de la pluie basée sur un mouvement de la main
- la capacité de changer l'affichage et les propriétés de simulation en cours d'utilisation de l'application
- la création d'une activité utilisant les modèles d'échelle de paysages existants.

Il faut noter que les développements actuellement sont stoppés par manque de financement.

1.2 Objectifs d'apprentissage scientifique

Le bac à sable permet de modéliser et comprendre des phénomènes environnementaux ainsi que leurs conséquences. La liste ci-dessous présente quelques applications pour lesquelles des activités pédagogiques utilisant une Sandbox ont déjà été développées :

- rupture de barrage
- crues et études de période de sécheresse
- effet de la pluie
- phénomène d'érosion et dépôt de sédiment
- principe de bassin versant
- mouvement de l'eau à la surface de la terre
- …

2. Matériel

Un bac à sable à réalité augmentée se compose des éléments listés ci-dessous. Chacun possède une fonction particulière et requiert des exigences pour permettre le fonctionnement du bac :

1. L'ordinateur exécute l'application logicielle
2. Le projecteur permet la projection de la surface topographique en couleur et de l'eau

3. La Kinect capte la topographie et les mouvements de l'utilisateur
4. Le logiciel permet d'interfacer la caméra et le projecteur, il calcule les écoulements et génère le rendu visuel. Il sera détaillé au Chapitre 3 .
5. Le bac à sable se compose
   a) Du bac
   b) De la potence permettant la fixation de la Kinect et du projecteur au-dessus du bac
   c) Du sable formant un relief assimilable à une topographie.

Il est donc intéressant, dans la suite de cette section, de présenter chaque élément individuellement et d'identifier les exigences qu'il devra remplir. La Figure 23 présente le bac à sable à réalité augmentée réalisé dans le cadre de ce travail et installé à l'université de Liège.

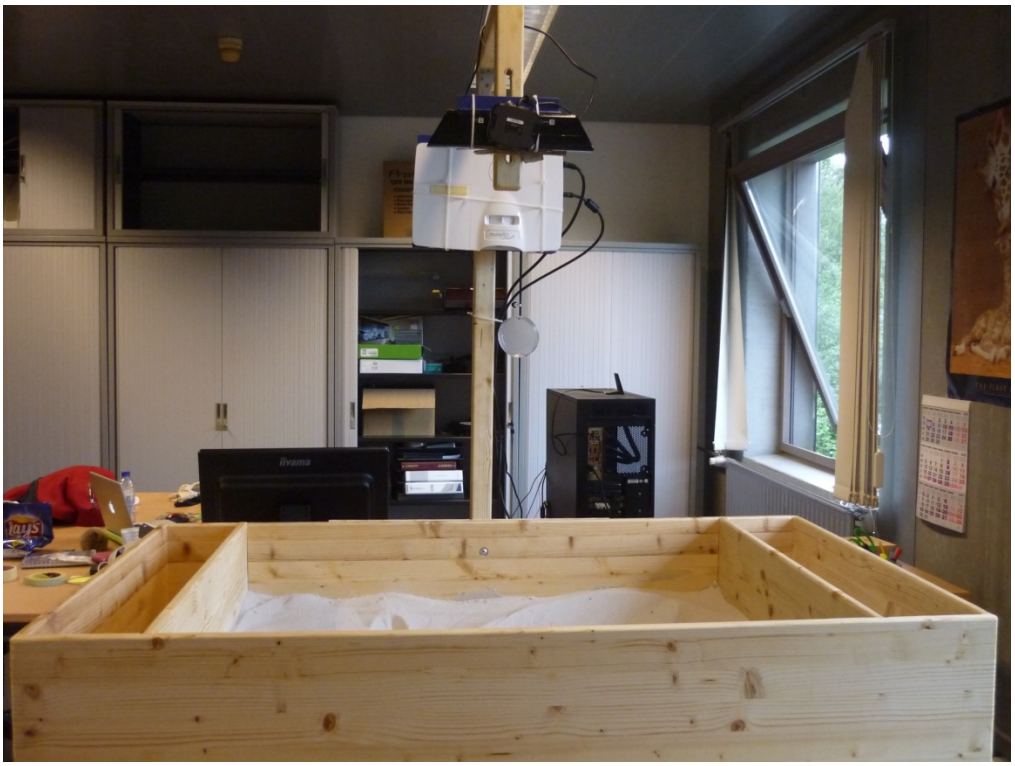


Figure 23: Bac à sable à réalité augmentée réalisé à l'Université de Liège

2.1 L'ordinateur

L'ordinateur s'interface avec la Kinect et le projecteur, il exécute le programme SARndbox qui gère le bac à sable à réalité augmentée.

Selon les spécifications du développeur, l'ordinateur d'un bac à sable à réalité augmentée doit être composé au minimum d'un CPU (Central Processing Unit) bi cœur (Intel Core i5 ou i7) cadencé à 3GHz (3G d'opérations effectuées à la seconde). Les besoins en mémoire RAM (Random Access Memory) et espace disque sont par contre peu exigeants : 2GB de RAM et 20 GB de disque dur sont largement suffisants. Une autre exigence essentielle est la présence d'un co-processeur graphique GPU (Graphics Processing Unit) et d'une carte graphique puissante de type Nvidia GeForce GTX 970

ou 1070. Cet ordinateur doit être sous une version Linux 64 bits avec la distribution de Linux Mate recommandée. Le programme SARndbox peut fonctionner en dualboot mais pas en virtual box. Une connexion internet n'est pas nécessaire pour le bon fonctionnement du logiciel. Le logiciel SARndbox peut être configuré pour démarrer automatiquement lorsque l'ordinateur démarre, dans ce cas, il n'a même besoin ni d'un moniteur, d'une souris ou d'un clavier. [13]

Ces exigences pour le GPU et CPU sont expliquées par le fait que les deux opérations logicielles qui occupent le plus de temps de calcul sont le rendu des cartes topographiques et le calcul de la simulation d'écoulement. Le premier fonctionne parfaitement avec un ordinateur standard ou moyennement évolué. Pour la seconde opération il est impératif de disposer d'une carte graphique puissante puisque le calcul d'écoulement est réalisé sur la carte graphique par l'intermédiaire de GSLS (OpenGL Shading Language) shader, cette procédure sera détaillée dans le Chapitre 3 -1.2 .

Comme le bac à sable sera installé à demeure dans un local alloué à un laboratoire numérique, le choix s'est porté sur un ordinateur fixe. La configuration détaillée de l'ordinateur est la suivante :

- Une tour avec la configuration présentée au Tableau 2
- Un écran/moniteur d'au moins 17 pouces
- Un clavier AZERTY
- Une souris.

PC ORDI.COM Série Pro-Intel Core i7 6700K 4.0gH/16GB/SSD 256 GB
Boitier Corsair Obsidian 750 D-fulltower
Alimentation corsair 750 Watts-Rmx-silencieuse-certification gold
Carte mère ASUS Maximus extreme – Chipset Z170
Processeur Intel Core I7 6700K 4.à GHz- 4 cœurs-8mb cache
Refroidisseur Corsair Vengeance LPX- 16 GB-DDR-4.3.200MHz (2*8GB) avec refroidisseur
SSD Samsung pro 850 256 GB
Carte graphique Asus ROG Strix GeForce GTX1070-8GB DDR-5- sorties 2*Displayport/DVI-D/2*HDMI
Lecteur/graveur CD/DVD-Samsung
Ecran llyama Prolite T2452MTS- 24 pouces-touchscreen-1920*1080 fullHD-2 msec-HDMI, DVI-D, 2HP intégrés
Clavier Logitech K120 avec fil et souris optique Logitech avec fil
Licence windows 10 pro française 64 bits
Préparation au labo avec installation utilitaires (anti-virus Ulg+ antispyware) +tests

Tableau 2: Configuration de l'ordinateur utilisé pour la Sandbox

2.2 La Kinect

La caméra 3D qui sert à capter le relief modelé dans le sable est un capteur Kinect de Microsoft. Elle sera disposée au-dessus du bac au milieu de la surface. Comme présenté dans le Chapitre 1 -2.1 , il existe 2 versions du capteur. Dans ce travail, une Kinect de chaque version a été utilisée :

- Kinect v1-Xbox 360 :
  - Numéro de modèle 1414
  - Utilisation prévue pour la mise en œuvre du bac dans la configuration existante proposée par le développeur du concept.
  - Connexion : USB 2.0
- Kinect v2 –Xbox One:
  - Utilisation prévue pour l'adaptation de la version existante à nouvelle version de caméra 3D.
  - Connexion : USB 3.0

Prévu comme accessoires de console de jeux vidéo avec des connecteurs spécifiques à une console de jeu Xbox, il est donc nécessaire, pour une utilisation sans celle-ci, de disposer de connecteurs vers un port USB spécifique avec une alimentation plus puissante alimentée à partir de 220 VAC.

2.3 Le projecteur

Le projecteur est utilisé pour illuminer la surface du sable avec la projection des surfaces topographiques colorées. Il est disposé dans l'axe médian et le long du bord le plus long du bac en hauteur.

Dans les spécifications d'installation, il est proposé un projecteur de la marque Benq MX631ST qui a un ratio de projection 4:3 pour correspondre au ratio de la Kinect avec une connexion via une prise VGA. Cette résolution est suffisante étant donné que la résolution de l'ensemble de bac à sable est limitée à 640 × 480 pixels ou 512 x 424 pixels pour la mesure de distance par la Kinect. Le projecteur doit avoir une courte focale pour permettre une faible distance entre la surface de projection et la lentille du projecteur et éviter qu'il soit disposé trop haut par rapport au bac. De plus, afin de permettre une projection optimale, il est nécessaire que le projecteur possède un bon ratio de contraste et une luminosité importante, pour permettre une utilisation du bac dans un espace avec une luminosité naturelle. [13]

Cependant, dans l'optique de pouvoir utiliser l'une ou l'autre version de la Kinect, il est nécessaire que le projecteur puisse avoir un ratio de projection variable, 4 :3 pour la Kinect première génération et 16:9 pour la seconde. Suite à cette exigence, le projecteur proposé par le développeur ne correspondait pas. La recherche du projecteur, s'est donc portée sur les caractéristiques suivantes :

- Focale courte (Rapport hauteur /largeur proche de 1)
- Luminosité élevée
- Ratio de projection réglable (4 :3 et 16 :9)
- Zoom

- Résolution HD

La caractéristique de zoom permet de positionner la Kinect et le projecteur à même hauteur au-dessus du bac et de régler l'image via un zoom numérique ou manuel.

La caractéristique de focale courte possède un inconvénient important. La plupart des projecteurs à focale courte projettent l'image au-dessus du niveau passant par le centre de la lentille comme présenté à la Figure 24. Cet inconvénient amène donc un décalage du projecteur par rapport au bac et un placement de celui-ci sur le côté du bac.

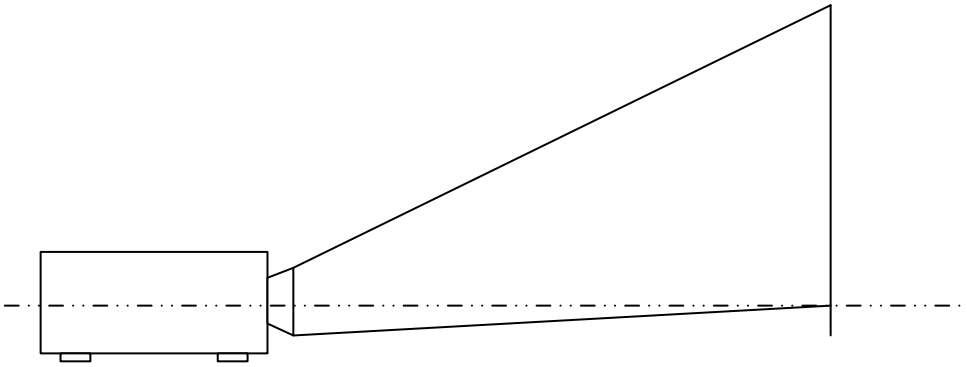


Figure 24: Décalage de l'image pour un projecteur avec une courte focale

Le projecteur choisi est donc le projecteur Optoma GT1080E qui sera connecté à l'ordinateur via un port HDMI, les caractéristiques principales sont présentées au Tableau 3. Les caractéristiques plus complètes et détaillées sont présentées à l'Annexe 3.

	Optoma GT1080E
Résolution	1920x 1080 pixels
Format d'affichage	16/9, 4/3
Luminosité	3000 ANSI lumens
Contraste	25 000 :1
Focale	Courte
Ratio de projection minimale	0,49
Ratio de projection maximale	1,00
Rapport de zoom	Fixe mais zoom numérique
Objectif	F=2.8, f=7,42 mm

Tableau 3: Caractéristiques principales du projecteur utilisé dans la Sandbox

2.4 Le bac à sable

2.4.a Le bac

Le bac contenant le sable doit avoir un ratio de dimension correspondant au ratio de la Kinect. Dans les spécifications de réalisation d'un bac, il est préconisé un bac de dimension 100x75cm avec la Kinect disposée à 1m de hauteur du sable, ce qui permet une résolution nominale de la caméra horizontal 1,56mm et verticale 2,79mm (plus ou moins le carré de résolution horizontale). Ces dimensions apportent un bon compromis entre hauteur de caméra et résolution suffisamment fine. [13]

Pour avoir la possibilité de passer d'une Kinect version 1 à la version 2, il fallait que la taille du bac à sable soit adaptable en fonction de la Kinect. Le choix des dimensions s'est basé sur la dimension du bac rectangulaire préconisé 1x0,75m. Afin de conserver la même position du projecteur et de la Kinect par rapport au bac. La dimension du bac pour la seconde Kinect est donc de 1,33 x0,75m. La hauteur des cotés est fixée à 30cm. Les photos à la Figure 25 présentent le bac réalisé dans le cadre de ce travail de fin d'études. Une solution de chariot à roulette avait été proposée mais la solution souhaitée était de poser le bac sur un bureau existant. Les plans de ce bac et du chariot sont disponibles à l'Annexe 4.

Figure 25 : Bac à sable à réalité augmentée : bac et potence

Le bac a été réalisé par nos soins en panneaux lamellés/collés de sapin blanc de 18 mm d'épaisseur pour les parois extérieures verticales et de multiplex 18mm d'épaisseur pour le fond. Afin de permettre un bac de dimension adaptable selon le choix de caméra, un bac de dimensions intérieure 1,33x0,75m a été réalisé. Dans les parois de 1,33m, deux coulisses ont été créées à 15 cm des bords pour y glisser des parois amovibles et réduire ainsi la dimension du bac. Le mode d'assemblage des

différents éléments a été réalisé par assemblage de type faux tenon et mortaise avec le système Domino Festool. On réalise dans les 2 pièces à assembler une mortaise et ensuite on insère un faux tenon en bois de hêtre qui sera collé dans les 2 mortaises. Des photos de cet assemblage sont disponibles à l'Annexe 5.

2.4.b La potence

La potence sert à disposer la caméra et le projecteur au-dessus du bac de manière à ce que leurs objectifs soient orientés vers l'intérieur du bac. Elle est visible sur la Figure 25.

Les dimensions de la potence, sont déterminées par la position des 2 appareils, sachant que la Kinect sera positionnée au centre du bac et que le projecteur sera fixé au long coté du bac en son milieu. Cependant, en raison du décalage de l'image du projecteur par rapport au centre de la lentille expliqué précédemment, un écartement de la potence de 11 cm par rapport au bac a dû être aménagé après essai. Cette disposition est visible dans les plans à l'Annexe 4. C'est cet écartement qui a conditionné les dimensions de la potence avec une hauteur à partir du bord supérieur du bac de 1m et une longueur de la partie transversale de 54cm. La position verticale des 2 appareils devait être adaptable pour permettre un champ visuel correspondant à l'intérieur du bac, ce qui dépend fortement de la quantité de sable dans le bac. Des mortaises traversantes sur une longueur de 25cm ont été réalisées afin d'ajuster parfaitement le positionnement des appareils lors des étapes de calibration de l'installation.

Afin d'assurer une certaine unité dans l'aspect du bac, cette potence a été réalisé en sapin blanc de section 4,7 x4,7 cm et d'équerres métalliques non renforcées au vu du faible poids de la Kinect en porte-à-faux (300g). Afin de pouvoir fixer la caméra et la Kinect, deux supports en acier soudé ont été réalisés pour poser les appareils et ensuite les attacher via des colliers de serrage en plastique ainsi que de l'autocollant double-face, pour ajuster leur position précisément. Un trou dans ces supports avec un boulon de 5,5cm de longueur passant dans les mortaises traversantes de la potence, permet de fixer ces éléments. Ces systèmes de fixation sont visibles à la Figure 26.

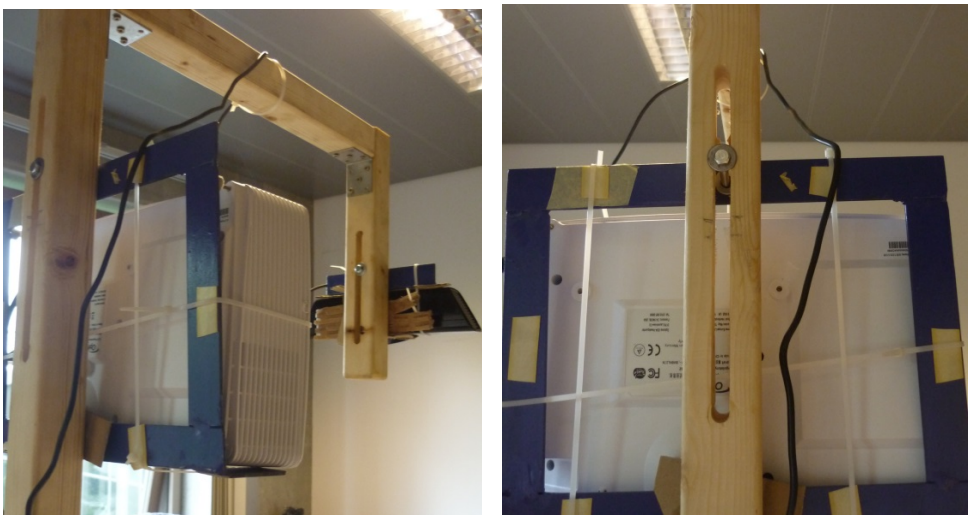


Figure 26: Fixation de la Kinect et du projecteur à la potence du bac à sable

2.4.c Le sable

Le sable dans le bac sert à modeler un relief qui sera assimilé à une topographie, ensuite captée par la Kinect. Dans les spécifications pour la construction d'un tel bac, une épaisseur de 10cm de sable dans le fond et un sable légèrement humide avec une granulométrie fine sont préconisés pour apporter une certaine tenue aux formes créées.

Il existe actuellement un autre de sable à l'aspect mouillé qui possède une bonne tenue sans devoir être humidifié, appelé sable cinétique (magique ou lunaire, suivant la marque). Il est composé de 98 % de sable naturel et d'additifs. Ceci lui permet d'être manipulé comme de la pâte à modeler, il peut d'une part s'écouler comme du sable sec et d'autre se maintenir en forme comme du sable mouillé compacté avec une tenue plus importante que dans le cas de sable normal. Il est souvent utilisé par les enfants dans le cadre d'activités pédagogiques. En plus de ses caractéristiques de maintien, ce sable est adhérent et donc occasionne peu de salissures et de pertes de sable lors des manipulations.

Cette solution paraissait être intéressante mais au vu de son cout élevé et du fait qu'il aurait fallu le faire importer, elle a été écartée. En effet, un poids de 150kg était nécessaire pour remplir le bac, le prix avoisinait 1600 euros contre 9 euros avec du sable normal de granulométrie comprise entre 1 et 0,1mm.

3. Le prix total des fournitures

Le Tableau 4 présente le prix total pour construire un tel bac à sable ainsi que deux variantes de prix selon l'utilisation d'une Kinect V2 ou d'une Kinect V1.

Matériel	Variante 1	Variante 2
Projecteur vidéo : Optoma GT1080E	857,03 €	857,03 €
Ordinateur	3000,00€	3000,00€
Câble HDMI	20,07€	20,07€
Kinect V1 (occasion)	30,00€	
Adaptateur Kinect V1 pour l'ordinateur	6,99 €	
Kinect V2		199,00€
Adaptateur Kinect V2 pour l'ordinateur		50,00€
Bac à sable (fournitures)	111,13€	111,13€
Sable	8,92€	8,92€
TOTAL=	**4036,14€**	**4248,15€**

Tableau 4 : Prix total du matériel pour le bac à sable à réalité augmentée

Chapitre 3 Le programme SARndbox

Le logiciel SARndbox est un programme Open Source gratuit sous licence publique générale GNU (GNU GPL) écrit en langage C++ et fonctionnant seulement sous Linux. Cette application est basée sur 2 autre logiciels développés aussi par l'université UC Davis de Californie : Vrui VR Toolkit et Kinect 3D Video Processing. En plus de l'utilisation de ces deux programmes, le logiciel fait appel à la bibliothèque logicielle graphique OpenGL qui permet de créer un rendu paramétrable en temps réel de la topographie captée par une Kinect avec une carte colorée et des lignes topographiques. SARndbox utilise aussi des GSLS shader pour calculer une simulation d'écoulement en temps réel.

Ce programme est relativement complexe, sa compréhension a représenté une part importante de ce travail. Avant de décrire la structure du programme, ce chapitre explique de manière brève les différents concepts nécessaires à la compréhension de cette partie informatique.

1. Concepts préliminaires

1.1 L'OpenGL

L'OpenGL est une bibliothèque logicielle utilisée principalement pour

- Créer un rendu 3D ou 2D dans une application fenêtrée
- Effectuer des calculs très rapides en utilisant la carte graphique et traitant une matrice donnée comme une image.

Cette seconde utilisation présente l'avantage d'un temps de calcul réduit par rapport à un calcul sur CPU.

Les fonctions OpenGL sont directement exécutées par la carte graphique à partir d'un contexte OpenGL déjà créé. Ce contexte peut être assimilé à un système de points, appelé vertex, qui correspond à une position dans un monde 3D à l'intérieur de l'application fenêtrée. Ces sommets sont alors liés entre eux par des formes géométriques comme des triangles, rectangles… Ensuite est appliqué un « coloriage » via une texture afin de permettre une représentation. [14]

Une texture est un objet OpenGL composé de pixels à afficher à l'écran. Un pixel contient 4 composantes couleur RGBA (Rouge, Green, Bleu, Alpha). La dernière composante est un quantificateur de la transparence. Une texture peut être vu comme une matrice où chaque pixel de la texture correspond à une cellule de la matrice [14]

La création d'un texture est toujours définie dans une fonction `init()` et passe par les étapes et les fonctions suivantes: [14]

1. Génération d'un identifiant de la texture

2. Verrouillage de l'objet : spécification la texture sur lequel on souhaite travailler

3. Application des filtres qui servent à traiter la qualité de l'image, comme par exemple un aspect lisse ou pixelisé.

4. Configurer la texture en donnant les données des pixels : on spécifie le nombre de composants couleur dans la texture, le largueur et la hauteur de la texture en pixels mais aussi le format de pixel. Le format de pixels correspond au mode de stockage des pixels. Dans format GL_RGBA, le pixel contient 4 valeurs de couleur différentes (R, G, B, A) et dans un format GL_LUMINANCE , le pixel contient 3 valeurs de couleur les mêmes et la dernières égalée à 0 (R, R, R,0). Dans cette phase, on spécifie aussi les données stockées, leur format (float, int…) et les données de ces pixels. Ces données du pixel seront fournies sous forme d'un pointeur contenant la matrice avec les valeurs des pixels imposés.

5. Déverrouillage de la texture

Les textures utilisées dans ce programme sont des textures avec l'extension ARB ce qui permet d'utiliser des dimensions de texture autre que des dimensions de puissances de 2 comme nécessaire dans le cas de texture sans l'extension ARB. Il faut savoir que toutes les cartes graphiques ne supportent des textures ARB.

Une fois définie, la texture peut être utilisée pour générer une représentation ou réaliser des calculs par la carte graphique. Cette étape dépend du mode de représentation choisi et fait intervenir une notion supplémentaire sur lequel il est utile de s'étendre.

L'utilisation d'une texture ne permet pas le stockage d'information, pour ces opérations il faut utiliser un objet OpenGL supplémentaire appelé Frame Buffer Object. Il est toujours composé de 3 types de frame, représenté à la Figure 27 :

- Color frame avec la couleur de chaque pixel, assimilable à une texture. Il peut en exister jusque 16 pour un même frame buffer object.
- Depth frame, contenant des données de profondeur pour chaque pixel
- Stencil frame pour filtrer les rendus à l'écran [14]

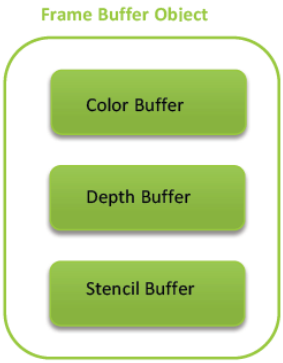


Figure 27: Composition d'un frame buffer object

Un frame est une zone mémoire de la carte graphique qui permet un stockage d'informations. [14]

La librairie permet aussi l'utilisation de double ou triple buffer. Ceci permet d'effectuer une opération lente sur l'un pendant que l'autre buffer est utilisé pour créer un rendu et lorsque que l'opération sur le premier est finie, les deux tâches sont échangées.

L'initialisation d'un FrameBufferObject suit les mêmes étapes que pour une texture et se fait à travers les étapes suivantes : [14]

1. Génération d'un identifiant du frame buffer object.

2. Verrouillage de l'objet : spécifie le frame buffer[2] sur lequel on souhaite travailler

Comme le frame buffer est un outil de stockage, il doit être lié à une texture pour permettre après une opération sur celle-ci qu'un résultat soit stocké dans ce frame buffer. Ce lien est effectué après l'initialisation de la texture et à la suite des opérations d'initialisation d'un frame buffer via la procédure suivante.

3. Attachement d'une texture à un frame buffer. Il se peut qu'une texture soit liée à deux frame buffer.

Suite à l'initialisation d'une texture et d'un frame buffer object avec le lien établis entre les 2 objets, ces objets peuvent alors être utilisés. Il existe beaucoup de manière d'utiliser ces textures et frame buffer, nous développerons ici le principe utilisé dans le cas du logiciel SARndbox soit les GSLS shaders qui traiteront les équations d'écoulement et les opérations de rendu purement graphique.

1.2 Les GSLS shaders

Un shader GSLS (OpenGLShading Language) est un programme exécuté par la carte graphique.

Avant de définir leur implémentation et utilisation, il est utile de comprendre le pipeline qui est la suite de opérations réalisées pour leur utilisation. Un schéma de ce pipeline est présenté à la Figure 28.

Après la définition des coordonnées des vertices, le vertex shader est compilé. Cette étape permet de réaliser des opérations sur les sommets et de valider les coordonnées des sommets. Ensuite, le fragment shader est compilé et celui-ci permet de réaliser des opérations sur les couleurs de chaque pixel de texture passée en argument et définie par les vertices. Après compilation, s'effectue l'édition de lien entre le vertex et le fragment shader. La séquence d'opérations se termine par l'exécution de ces shaders, un résultat ou des valeurs peuvent alors être stockées dans un frame buffer object. [15]

[2] Dans la suite de ce travail, le terme frame buffer sera utilisé pour parler de frame buffer objects

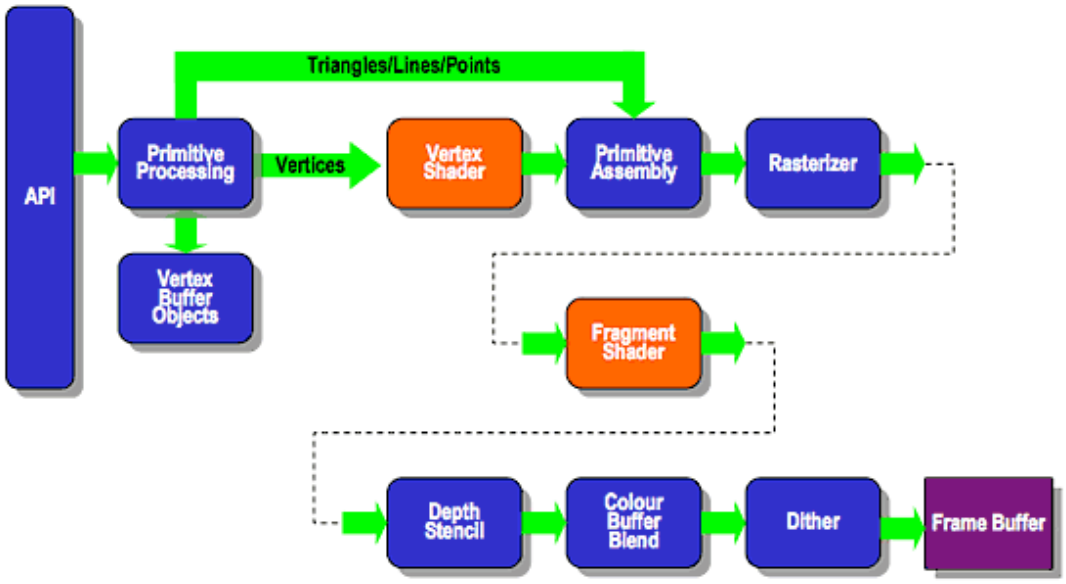


Figure 28 : Pipeline d'utilisation d'un shader [15]

Le vertex et fragment shader se présente sous forme de fichier, respectivement avec l'extension « .vs » et « .fs. » Ces fichiers seront donc appelés dans le programme principal via une étape d'initialisation et le lien vers les shader sera spécifié avec des instructions spéciales dans le makefile.

L'initialisation d'un shader se fait dans la méthode OpenGL `init()` à la suite des opérations d'initialisation de textures et frame buffer object ainsi que celle d'attachement de texture et frame buffer.

L'initialisation d'un shader comporte les étapes suivantes : [15]

1. Spécification du vertex shader à compiler
2. Spécification du fragment shader à compiler
3. Edition des liens entre le vertex et fragment shader, pour la création du programme à utiliser
4. Suppression des vertex et fragments shader car l'édition du lien a été effectuée.
5. Spécification de la localisation de chaque variable uniforme entrée en paramètres dans un programme de shader.

Après l'initialisation, dans une fonction `Init()`, l'utilisation d'un shader se fait en différentes étapes et utilise les textures et frame buffer. Ces étapes sont les suivantes : [15]

1. Verrouillage du frame buffer pour spécifier que le résultat va être stocké dans ce frame buffer

2. Spécification du color buffer choisi pour stocker des informations

3. Définition des coordonnées de la texture définissant le rectangle de pixels sur lequel est appliqué le shader

4. Installation du programme pour créer un rendu

5. Spécification des valeurs des variables uniformes utilisées en paramètres de l'utilisation d'un shader. Ces valeurs passées en argument doivent être des objets OpenGL soit des variables simples, soit des textures

6. Délimitation des sommets en pixels sur lesquels vont être réalisées ces opérations.

7. Exécution du code du shader.

Le développement du fragment et du vertex shader sont issus d'un langage propre (GSLS) avec les types de variables et les opérations différentes. Dans le fragment shader, on retrouve un code qui est exécuté pour chaque pixel se trouvant à l'intérieur des vertices définis lors de l'exécution. Pour obtenir des informations contenues dans une texture qui seront exploitables comme une matrice, il faut accéder à la valeur d'un pixel couleur (R, G,B ou A) et effectuer des opérations sur les données ce qui sera le cas pour le calcul d'écoulement ou le rendu graphique. Le résultat de ces opérations est stocké dans les frame buffer définis en début d'utilisation du shader.

Ces frame buffer contenant les résultats étant des objets stockés sur le GPU, une opération de transfert vers le CPU sera nécessaire pour pourvoir obtenir les informations dans un format exploitable comme une matrice ou un vecteur. [15]

Il est utile de mentionner que les fonctions `display()` et `init()` sont des fonctions de type standard en OpenGL et qui seront exécutées automatiquement et en boucle sans nécessairement être appelées explicitement dans le code.

1.3 La structure d'un programme C++

Un programme en langage C++ est composé de 2 types de fichier :

- Des fichiers header avec une extension « .h », qui contiennent la définition des variables principales et le prototype des fonctions utilisées.
- Des fichiers sources avec une extension « .cpp », qui contiennent le code composé d'instructions pour chacune des fonctions définies dans le header portant le même nom.

Le C++ est un langage orienté objet qui repose sur la notion de classes. La classe est un objet informatique qui possède des attributs et des méthodes. Chaque élément est défini comme appartenant à une classe et hérite des attributs et méthodes. Les attributs sont les variables contenues dans une classe et une méthode est une fonction qui peut être opérée sur les attributs. [16]

L'utilisation d'une classe passe d'abord par deux étapes préliminaires : la création d'une classe et l'initialisation des attributs. [16]

Une classe est définie dans un header. Ensuite, l'initialisation de cette classe est réalisée à partir d'une méthode de la classe spéciale appelée constructeur. Il porte le même nom que la classe et n'a aucun type retour, mais peut avoir des arguments. Après l'initialisation de la classe, on peut alors accéder à la valeur d'un attribut d'une classe ou exécuter une méthode. [16]

Pour produire un exécutable à partir de nombreux fichiers sources et librairies, on utilise un fichier Makefile qui examine les dépendances des fichiers sources et des librairies, qui ensuite appelle, avec les options sélectionnées, le compilateur et l'éditeur de liens.

Les opérations d'édition de lien et de compilation font parties du processus de génération d'un exécutable. Ce processus est représenté sur le schéma à la Figure 29. Ce processus commence par le préprocesseur qui vérifie les fichiers d'entête ou headers écrits par le développeur. L'étape de compilation transforme les fichiers source en code binaire sous forme de fichier objet. Ensuite, l'édition de lien assemble tous les fichiers objets qui constituent un programme et créent un fichier exécutable. Ce fichier exécutable peut alors être directement exécuté dans un terminal qui ouvrira directement l'application fenêtrée.

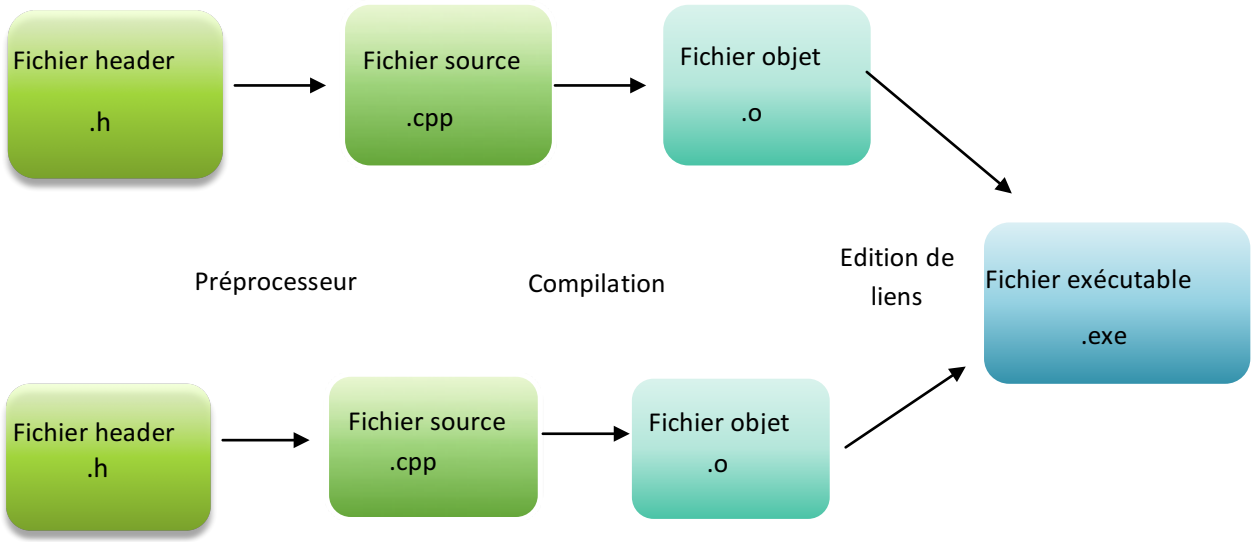


Figure 29: Schéma de création d'un exécutable

Ce processus est réalisé en configurant un fichier makefile, qui est ensuite lancé dans un terminal via la commande `make`.

2. La structure du programme

Comme mentionné précédemment, le logiciel est basé sur 2 autres librairies développées à l'université UC Davis de Californie : [13]

- Vrui VR Toolkit, qui est une suite d'outils, fonctions et de taches simplifiant la création d'application de réalité virtuelle et augmentée.
- Kinect 3D Video Processing qui permet quand une Kinect est connectée à un ordinateur, de l'utiliser comme caméra 3D, de créer des représentations 3D et d'être utilisée dans des applications de réalité virtuelle.

Le premier logiciel est présenté sous forme d'une librairie indépendante. Cet outil de réalité virtuelle répond à trois objectifs principaux :

- une indépendance vis-à-vis de système d'affichage (écran, projecteur)
- une indépendance vis-à-vis de la distribution (nombre d'ordinateur, type de connexion…)
- une adaptation à plusieurs type d'outils utilisés dans la réalité virtuelle (Kinect, gant capteur de donnée, outil de traque…).

Le logiciel SARndbox n'utilise qu'une partie des nombreuses fonctionnalités offertes par cette librairie [13]

Le second programme s'organise autour de la définition d'une grande quantité de classes et d'une application simple. Les classes définies traitent : [13]

- Des aspects de la librairie Libusb qui gère les connexions à l'ordinateur via tout type de port Usb
- Le contrôle de la Kinect et donc la caméra couleur et la mesure de profondeur
- Des opérations utilisées pour projeter une image de profondeur et de couleur dans un espace 3D.

En plus de ces deux librairies le programme SARndbox utilise les librairies C++ standard et les librairies sous Linux implémentées par le développeur, M. Kreylos.

Le programme SARndbox est fonctionnel seulement sur Linux. Cela est dû au fait que le logiciel Vrui VR Toolkit n'est pas compatible Windows à cause de la construction de certaines template C++ présents dans le logiciel. De plus, certaines parties de l'architecture logicielle du logiciel Vrui VR Toolkit sont implémentées grâce à des fonctionnalités spécifiques à Unix. [13]

Le logiciel est Open Source. Il est donc disponible en téléchargement gratuit. Suite à ce téléchargement, l'utilisateur est libre de modifier ce programme et puis le remettre en distribution libre sur la toile. Le dossier de téléchargement possède plusieurs types de fichiers :

- Fichiers sources
- Fichiers header
- Makefile
- Des shaders
- Des fichiers de configurations établis lors de l'installation du logiciel

Le Tableau 5 présente la liste des fichiers source et header composant le programme avec leur fonction principale.

Nom	.h	.cpp	Fonction
BathymetrySaverTool	X	X	Outil pour sauver dans un fichier la bathymétrie courante utilisée dans la Sandbox
CalibrateProjector	X	X	Utilitaire pour calculer les transformations de calibration du projecteur dans l'espace 3D capté par la Kinect
Config	X		Configuration des fichiers header pour la Sandbox
DEM	X	X	Classe pour représenter un modèle d'élévation digitale (DEM) comme une texture contenant les valeurs exprimées en variable de type float
DEMTool	X	X	Classe outil pour charger un fichier dans le programme et colorier la surface de sable basé sur la distance présente dans le fichier DEM
DepthImageRenderer	X	X	Classe pour centraliser le stockage des frames bruts captés par la Kinect et les frames filtrés sur le GPU ainsi que pour exécuter les tâches répétitives de rendus
ElevationColorMap	X	X	Classe pour représenter la carte topographique en couleur à partir d'une carte topographique
FindBlobs	X		Fonction helper pour extraire des pixels possédant une caractéristique particulière
FrameFilter	X	X	Classe pour filtrer les frames de profondeur arrivant de la Kinect avec un code pour détecter des valeurs instables dans chaque pixel et compléter les pixels sans valeurs à partir d'échantillon valides
GlobalWaterTool	X	X	Classe pour ajouter ou retirer globalement de l'eau sur la Sandbox
HandExtractor	X	X	Classe pour identifier une main dans une image de profondeur
LocalWaterTool	X	X	Classe outil pour ajouter ou retirer localement de l'eau sur la Sandbox
RainMaker	X	X	Classe pour détecter un objet qui bouge à travers un intervalle de profondeur sur une image de profondeur et pour déclencher une chute de pluie sur le terrain virtuel
Sandbox	X	X	Application de réalité virtuelle pour gérer la Sandbox
ShaderHelper	X	X	Fonction header pour créer des GSLS shader à partir de fichiers textes
SurfaceRenderer	X	X	Classe pour rendre une surface définie par une grille régulière dans l'espace caméra
Types	X		Déclaration des types de données échangées dans les modules du logiciel
WaterRenderer	X	X	Classe pour rendre une surface d'eau définie à partir d'une bathymétrie et des valeurs du niveau d'eau

WaterTable2	X	X	Classe pour simuler l'écoulement d'eau utilisant une simulation basée sur les équations de Saint Venant

Tableau 5 : Liste des fichiers contenus dans le dossier de téléchargement du logiciel SARndbox

L'insertion et l'appel de fonctions de ces fichiers, permettront lors de la compilation et de l'édition de liens de générer les fichiers objets et deux exécutables :

- L'application SARndbox
- L'application de calibration du projecteur et de la Kinect.

2.1 Analyse du programme principal

Afin de comprendre comment fonctionne le programme de manière globale, ce chapitre présente le fichier principal qui gère le programme, soit le fichier `Sandbox.cpp`.

Il implémente les fonctionnalités suivantes :

- gestion du rendu graphique et utilisation des fonctions OpenGL
- interaction avec l'utilisateur, création des menus et fenêtres dans l'application
- diverses fonctions utilisées dans la boucle principale et appelant d'autres fonctions ou librairies
- la fonction principale pour l'initialisation l'application et des données transitées.

Les paragraphes suivants détaillent la fonction principale qui constitue le main de l'exécutable de l'application. La Figure 31 présente un schéma bloc de cette boucle principale ainsi que les appels au programme Kinect. La légende de cette figure est disponible à la Figure 30. L'interaction avec le programme VRui ne sera pas présentée dans ce schéma car il est utilisé dans presque tous les appels, en effet tous les objets créés sont basés sur des librairies implémentées dans cet outil.

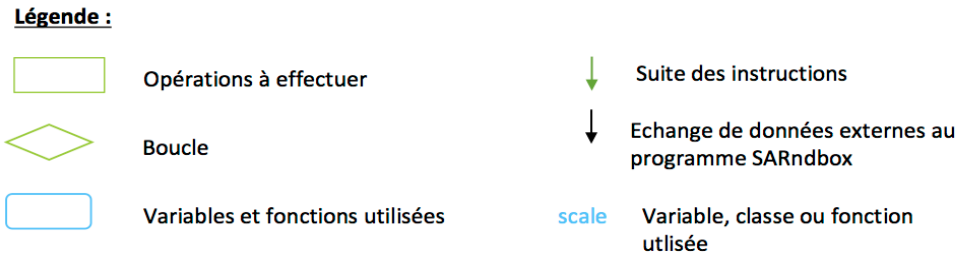
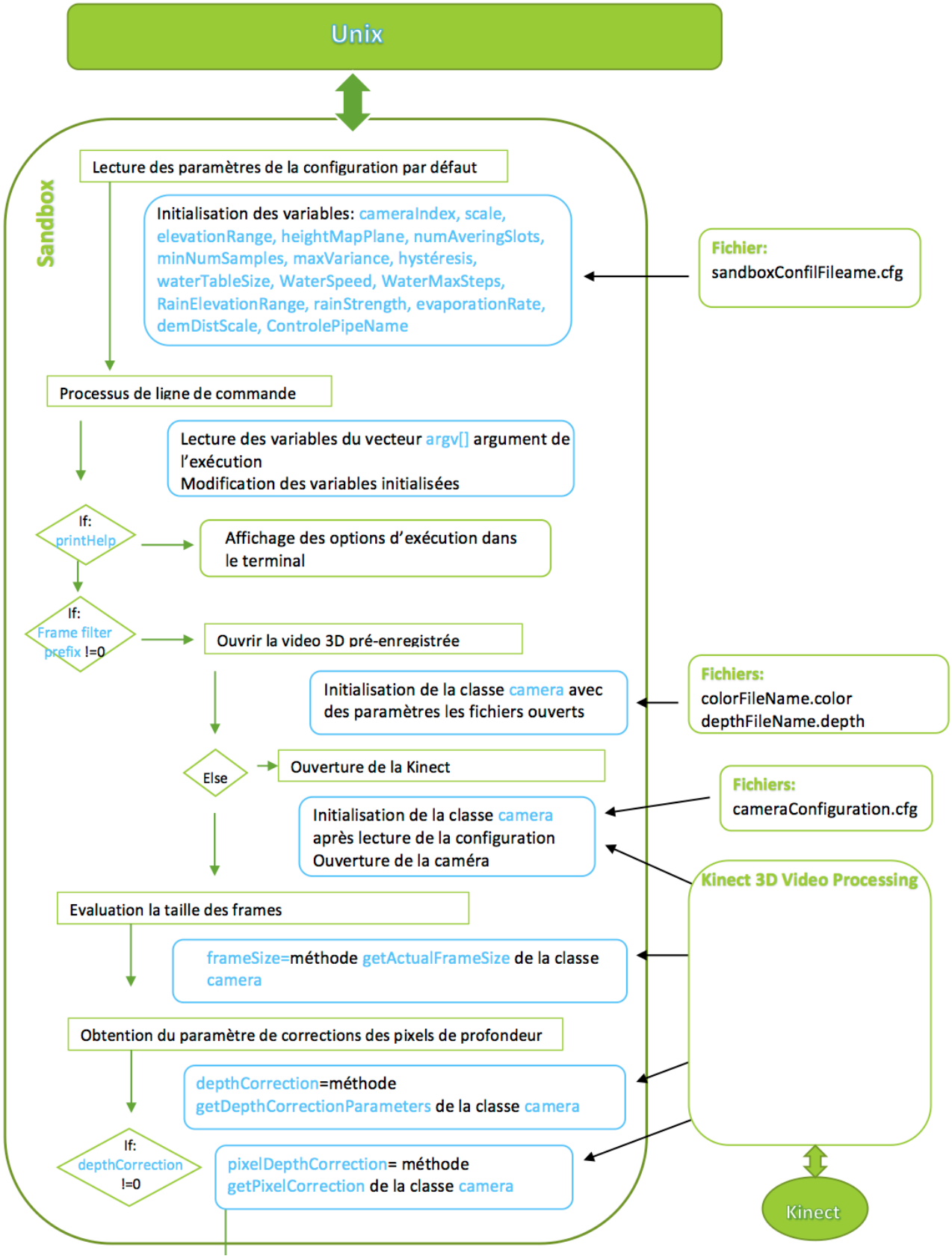
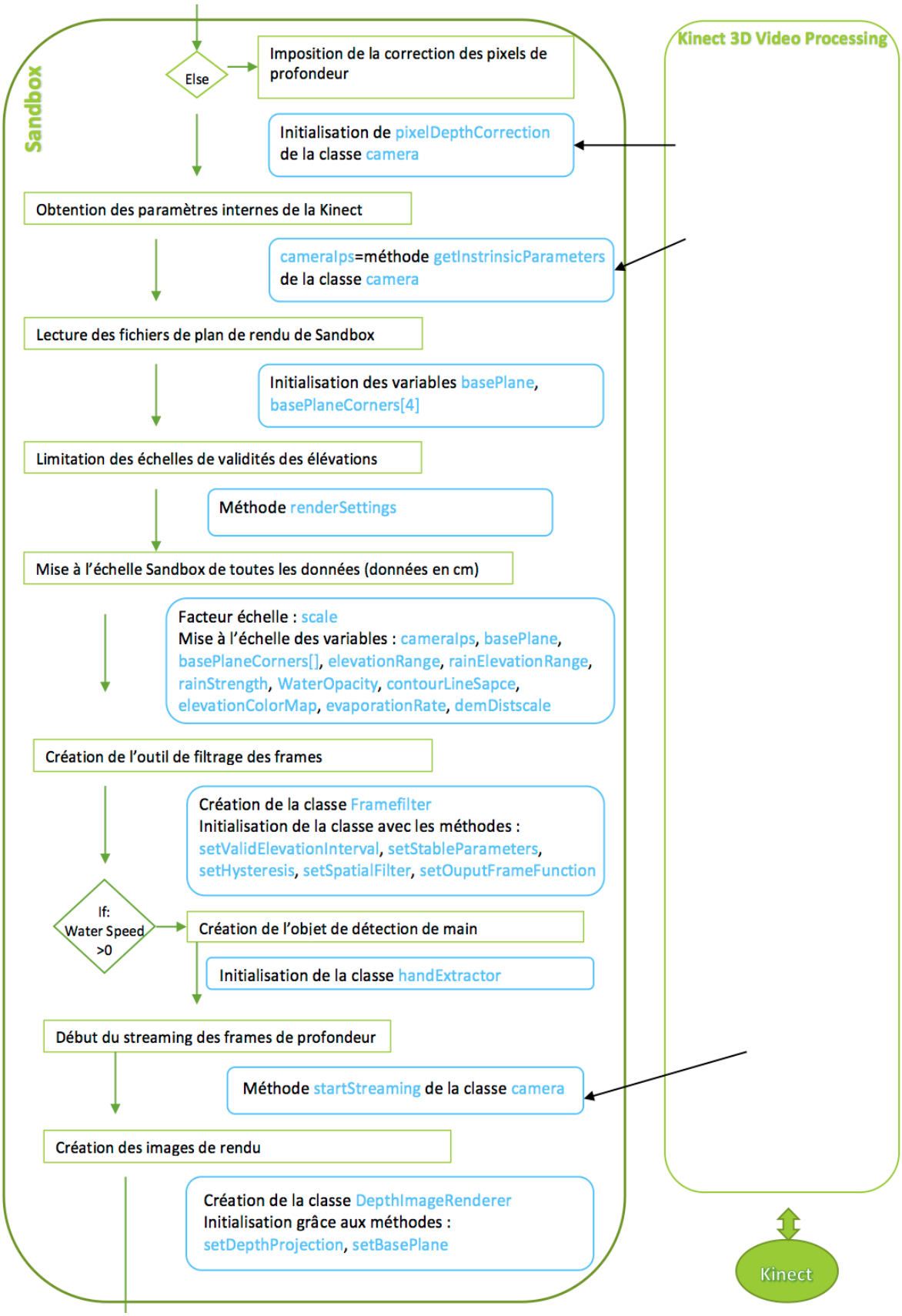


Figure 30: Légende du schéma bloc de la boucle principale du logiciel SARndbox


Unix
Sandbox
Lecture des paramètres de la configuration par défaut
Initialisation des variables: cameraIndex, scale, elevationRange, heightMapPlane, numAveringSlots, minNumSamples, maxVariance, hystéresis, waterTableSize, WaterSpeed, WaterMaxSteps, RainElevationRange, rainStrength, evaporationRate, demDistScale, ControlePipeName
Fichier:
sandboxConfilFileame.cfg
Processus de ligne de commande
Lecture des variables du vecteur argv[] argument de l'exécution
Modification des variables initialisées
If: printHelp
Affichage des options d'exécution dans le terminal
If: Frame filter prefix !=0
Ouvrir la video 3D pré-enregistrée
Initialisation de la classe camera avec des paramètres les fichiers ouverts
Fichiers:
colorFileName.color
depthFileName.depth
Else
Ouverture de la Kinect
Initialisation de la classe camera après lecture de la configuration
Ouverture de la caméra
Fichiers:
cameraConfiguration.cfg
Kinect 3D Video Processing
Evaluation la taille des frames
frameSize=méthode getActualFrameSize de la classe camera
Obtention du paramètre de corrections des pixels de profondeur
depthCorrection=méthode getDepthCorrectionParameters de la classe camera
If: depthCorrection !=0
pixelDepthCorrection= méthode getPixelCorrection de la classe camera
Kinect


Sandbox
Else
Imposition de la correction des pixels de profondeur
Initialisation de pixelDepthCorrection de la classe camera
Obtention des paramètres internes de la Kinect
cameraIps=méthode getInstrinsicParameters de la classe camera
Lecture des fichiers de plan de rendu de Sandbox
Initialisation des variables basePlane, basePlaneCorners[4]
Limitation des échelles de validités des élévations
Méthode renderSettings
Mise à l'échelle Sandbox de toutes les données (données en cm)
Facteur échelle : scale
Mise à l'échelle des variables : cameraIps, basePlane, basePlaneCorners[], elevationRange, rainElevationRange, rainStrength, WaterOpacity, contourLineSapce, elevationColorMap, evaporationRate, demDistscale
Création de l'outil de filtrage des frames
Création de la classe Framefilter
Initialisation de la classe avec les méthodes : setValidElevationInterval, setStableParameters, setHysteresis, setSpatialFilter, setOuputFrameFunction
If: Water Speed >0
Création de l'objet de détection de main
Initialisation de la classe handExtractor
Début du streaming des frames de profondeur
Méthode startStreaming de la classe camera
Création des images de rendu
Création de la classe DepthImageRenderer
Initialisation grâce aux méthodes : setDepthProjection, setBasePlane
Kinect 3D Video Processing
Kinect

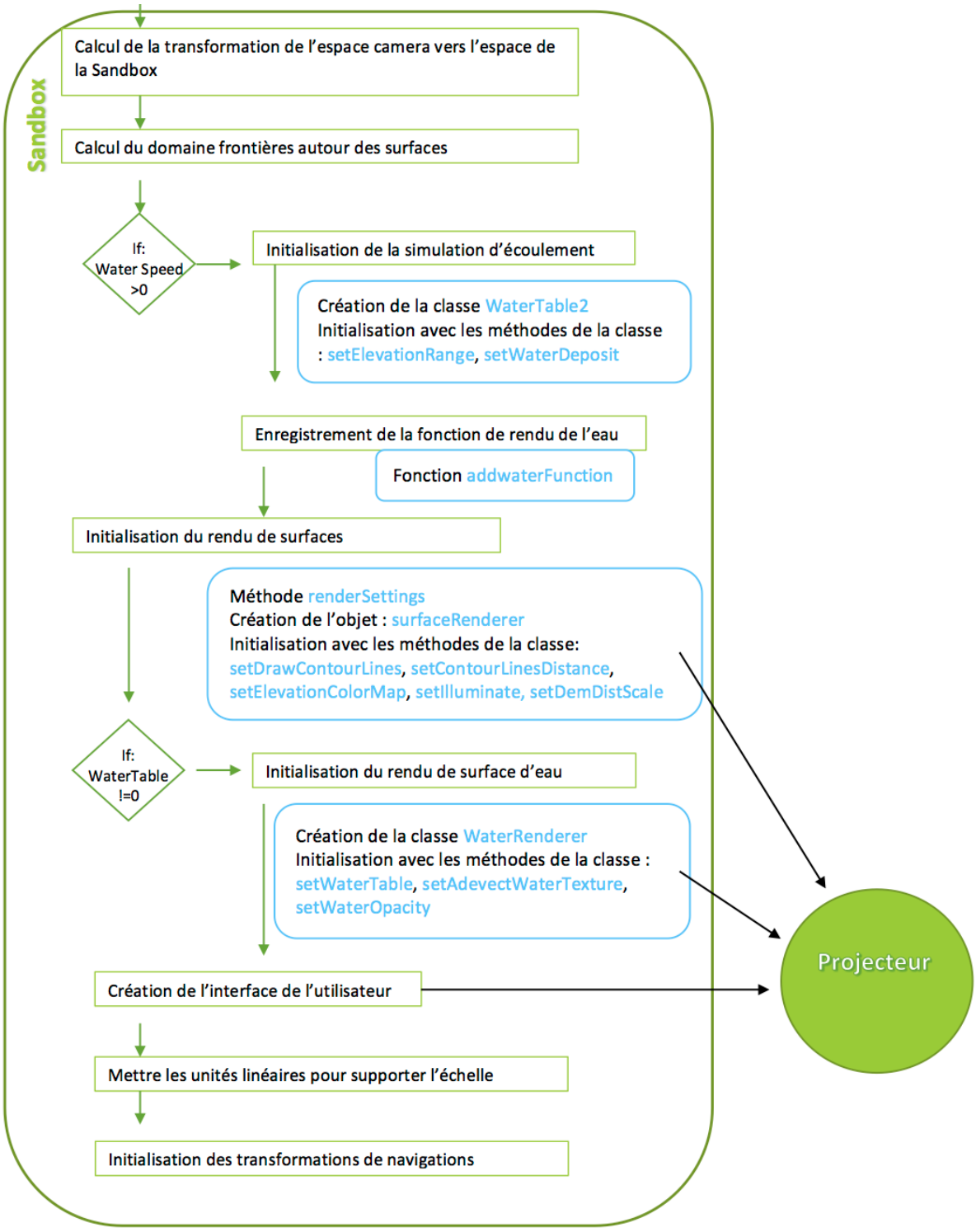


Figure 31: Schéma Bloc de la boucle principale du logiciel SARndbox

Il est à noter que ce programme est très compliqué et qu'en matière de documentation, il n'existe pas de documents de spécification du logiciel, seul le code est commenté.

Une partie non négligeable de ce travail, de surcroit pour une non informaticienne, fut de comprendre le code, le langage utilisé et comment le programme fonctionne afin de pouvoir le modifier. Ce rapport ne contient d'ailleurs que les éléments les plus spécifiques du logiciel.

2.2 Analyse de la captation d'un relief par la Kinect

Suite aux instructions propres à la librairie créée par le développeur pour interagir avec la Kinect, les images de la caméra couleur et infrarouge sont obtenues à une fréquence de 30 Hz. Ces images sont obtenues sur la forme d'un frame buffer object composé d'un frame couleur et d'un frame profondeur. Comme expliqué au Chapitre 3 -1, chaque frame est composé de pixels contenant des données. Dans le cas du frame couleur, chaque donnée de pixel contient trois composantes : rouge, vert, bleu (RGB), tandis que dans le cas du frame de profondeur la donnée est unique et correspond à la distance du pixel.

Dans le cas de la Sandbox, seul le frame de profondeur est utilisé. Les distances entre la surface captée seront exprimées en données de type `float`. La fonction `getdata()` présente dans le programme permet l'opération de transformation du frame buffer en type `float` .

A cette opération s'ajoute une opération de filtrage des frames bruts venant de la Kinect et de transformation en frame alors exploitables pour la représentation et le calcul de simulation. Le filtrage de frame est traité dans le fichier `FrameFilter.cpp`. Il est constitué d'un lissage temporel complexe effectué sur 30 frames bruts à la fois soit toutes les secondes à partir de frames de comparaison définis dans le programme. Cette opération est effectuée par un parcours des valeurs du frame brut et permet alors d'identifier et de remplacer la valeur des pixels sans valeurs ou ceux dont la valeur est incohérente par rapport aux pixels adjacents.

Un lissage spatial est aussi effectué sur les valeurs obtenues suite au lissage temporel, cette opération modifie faiblement les valeurs des pixels.

Une transformation du frame final est alors effectuée pour être dans un format directement représentable et utilisable pour les simulations.

3. Simulation d'écoulement

Afin de comprendre comment a été géré l'écoulement dans le programme SARndbox, ce point présente d'abord le modèle mathématique utilisé et comment il est résolu numériquement.

3.1 Modèle mathématique

La simulation d'écoulement est basée sur la résolution numérique les équations de Saint Venant en deux dimensions, soit les équations en eaux peu profondes. Ces équations sont une version intégrée sur la hauteur des équations de Navier-Stokes concernant les écoulements fluides. Ce système d'équations présenté à l'Équation 3.1 Les inconnues de ce système sont (h, hu, hv).

$$\begin{cases} \dfrac{\partial h}{\partial t} + \dfrac{\partial (hu)}{\partial x} + \dfrac{\partial (hv)}{\partial y} = 0 \\ \dfrac{\partial (hu)}{\partial t} + \dfrac{\partial \left(hu^2 + \frac{1}{2}gh^2\right)}{\partial x} + \dfrac{\partial (huv)}{\partial y} = -gh\dfrac{\partial (B)}{\partial x} \\ \dfrac{\partial (hv)}{\partial t} + \dfrac{\partial (huv)}{\partial x} + \dfrac{\partial \left(hv^2 + \frac{1}{2}gh^2\right)}{\partial y} = -gh\dfrac{\partial (B)}{\partial y} \end{cases}$$

Équation 3.1

Avec h, la hauteur du fluide au dessus du fond, u et v les vitesses du fluide, respectivement dans les directions x et y, g l'accélération gravitationnelle et $B(x, y)$ l'élévation du fond.

L'obtention de ce système à partir des équations de Navier-Stokes, repose sur les hypothèses suivantes :

- Le fluide est de l'eau supposée incompressible et de densité constante
- La distribution de pression est hydrostatique
- La pente de fond est faible
- La vitesse sur la section du canal est repartie de manière uniforme
- Les pertes de charge en flux instationnaires peuvent être simulées en utilisant une formule de résistance « steady state »
- On ne considère aucune pente de frottement
- On ne considère aucun apport latéral d'eau
- Le canal a une section uniforme dans le temps.

Le système peut être ré-écrit sous la forme de l'Équation 3.2 en imposant la variable $w = h + B$, la hauteur de la surface libre.

$$\begin{cases} \dfrac{\partial w}{\partial t} + \dfrac{\partial (hu)}{\partial x} + \dfrac{\partial (hv)}{\partial y} = 0 \\ \dfrac{\partial (hu)}{\partial t} + \dfrac{\partial \left(\frac{(hu)^2}{w-B} + \frac{1}{2}g(w-B)^2\right)}{\partial x} + \dfrac{\partial \left(\frac{(hu)(hv)}{w-B}\right)}{\partial y} = -g(w-B)\dfrac{\partial (B)}{\partial x} \\ \dfrac{\partial (hv)}{\partial t} + \dfrac{\partial \left(\frac{(hu)(hv)}{w-B}\right)}{\partial x} + \dfrac{\partial \left(\frac{(hv)^2}{w-B} + \frac{1}{2}g(w-B)^2\right)}{\partial y} = -g(w-B)\dfrac{\partial (B)}{\partial y} \end{cases}$$

Équation 3.2

Les inconnues de ce système sont donc $(w, hu, hv)^T$

3.2 Schéma de résolution numérique

3.2.a Schéma de discrétisation spatiale

La résolution numérique des équations est basée sur un schéma du second ordre « well-balanced » «central upwind » développé par Alexander Kurganov et Guergana Petrova. Ce schéma de résolution assure que les propriétés suivantes sont satisfaites :

- Schéma « well-balanced »
- Conservation d'une hauteur d'eau positive en tout temps

En même temps, ce schéma reste suffisamment précis, efficace et robuste. [17]

Le système d'équation peut être écrit sous la forme suivante :

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial\big(\boldsymbol{F}(\boldsymbol{U},B)\big)}{\partial x} + \frac{\partial\big(\boldsymbol{G}(\boldsymbol{U},B)\big)}{\partial x} = S(U,B)$$

Équation 3.3

Avec :

- Le terme inertiel :

$$\boldsymbol{U} = (w, hu, hv)^T$$

Équation 3.4

- Les termes convectifs :

$$\boldsymbol{F}(\boldsymbol{U},B) = \left(hu, \frac{(hu)^2}{w-B} + \frac{1}{2}g(w-B)^2, \frac{(hu)(hv)}{w-B}\right)^{\boldsymbol{T}}$$

Équation 3.5

$$\boldsymbol{G}(\boldsymbol{U},B) = \left(hv, \frac{(hu)(hv)}{w-B}, \frac{(hv)^2}{w-B} + \frac{1}{2}g(w-B)^2\right)^{\boldsymbol{T}}$$

Équation 3.6

- Le terme source :

$$\boldsymbol{S}(\boldsymbol{U},\boldsymbol{B}) = \left(0, -g(w-B)\frac{\partial(B)}{\partial x}, -g(w-B)\frac{\partial(B)}{\partial y}\right)^{\boldsymbol{T}}$$

Équation 3.7

Une discrétisation spatiale de type volumes finis des Équations 3-4-3.7 permet d'exprimer les équations sous la forme de l'Équation 3.8.

$$\frac{d}{dt}\overline{\boldsymbol{U}}_{j,k}(t) = -\frac{\boldsymbol{H}^x_{j+\frac{1}{2},k}(t) - \boldsymbol{H}^x_{j-\frac{1}{2},k}(t)}{\Delta \mathrm{x}} - \frac{\boldsymbol{H}^y_{j,k+\frac{1}{2}}(t) - \boldsymbol{H}^y_{j,k-\frac{1}{2}}(t)}{\Delta \mathrm{y}} + \overline{\boldsymbol{S}}_{j,k}(t)$$

Équation 3.8

Avec $C_{j,k} \coloneqq \left[x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}\right] \times \left[y_{k-\frac{1}{2}}, y_{k+\frac{1}{2}}\right]$, la cellule sur laquelle est effectuée le calcul.

Le terme $\overline{\boldsymbol{U}}_{j,k}(t)$ représente la moyenne sur la cellule de la variable $\boldsymbol{U}$:

$$\overline{\boldsymbol{U}}_{j,k}(t) \approx \frac{1}{\Delta x \Delta y} \iint_{C_{j,k}} \boldsymbol{U}(x,y,t)\, dx\, dy$$

Équation 3.9

L'Équation 3.8 peut alors être résolue grâce à un schéma d'intégration temporelle stable. [17]

Les flux numériques $\boldsymbol{H}^x$ et $\boldsymbol{H}^y$ peuvent être calculés à l'aide de L'Équation 3.10:

$$\boldsymbol{H}^x_{j+\frac{1}{2},k} = \frac{a^+_{j+\frac{1}{2},k}\boldsymbol{F}\left(\boldsymbol{U}^E_{j,k}, B_{j+\frac{1}{2},k}\right) - a^-_{j+\frac{1}{2},k}\boldsymbol{F}\left(\boldsymbol{U}^W_{j+1,k}, B_{j+\frac{1}{2},k}\right)}{a^+_{j+\frac{1}{2},k} - a^-_{j+\frac{1}{2},k}}$$

$$+ \frac{a^+_{j+\frac{1}{2},k} a^-_{j+\frac{1}{2},k}}{a^+_{j+\frac{1}{2},k} - a^-_{j+\frac{1}{2},k}}\left[\boldsymbol{U}^W_{j+1,k} - \boldsymbol{U}^E_{j,k}\right]$$

$$\boldsymbol{H}^y_{j,k+\frac{1}{2}} = \frac{b^+_{j,k+\frac{1}{2}}\boldsymbol{G}\left(\boldsymbol{U}^N_{j,k}, B_{j,k+\frac{1}{2}}\right) - b^-_{j,k+\frac{1}{2}}\boldsymbol{G}\left(\boldsymbol{U}^S_{j,k+1}, B_{j,k+\frac{1}{2}}\right)}{b^+_{j,k+\frac{1}{2}} - b^-_{j,k+\frac{1}{2}}}$$

$$+ \frac{b^+_{j,k+\frac{1}{2}} b^-_{j,k+\frac{1}{2}}}{b^+_{j,k+\frac{1}{2}} - b^-_{j,k+\frac{1}{2}}}\left[\boldsymbol{U}^S_{j,k+1} - \boldsymbol{U}^N_{j,k}\right]$$

Équation 3.10

Avec, $B_{j,k+\frac{1}{2}}$ l'approximation de l'élévation du fond et $\boldsymbol{U}^{E,W,N,S}_{j,k}$ la valeur de la reconstruction linéaire de $\boldsymbol{U}$ sur le contour de la cellule $C_{j,k}$, définis par les expressions 3.11-3.14 : [17]

$$\boldsymbol{U}^E_{j,k} = \overline{\boldsymbol{U}}_{j,k}(t) + \frac{\Delta x}{2}\frac{\partial \boldsymbol{U}_{j,k}}{\partial x}$$

Équation 3.11

$$\boldsymbol{U}^W_{j,k} = \overline{\boldsymbol{U}}_{j,k}(t) - \frac{\Delta x}{2}\frac{\partial \boldsymbol{U}_{j,k}}{\partial x}$$

Équation 3.12

$$\boldsymbol{U}^N_{j,k} = \overline{\boldsymbol{U}}_{j,k}(t) + \frac{\Delta y}{2}\frac{\partial \boldsymbol{U}_{j,k}}{\partial y}$$

Équation 3.13

$$\boldsymbol{U}^S_{j,k} = \overline{\boldsymbol{U}}_{j,k}(t) - \frac{\Delta y}{2}\frac{\partial \boldsymbol{U}_{j,k}}{\partial y}$$

Équation 3.14

Les vitesses locales au droit de chaque côté de la cellule $b^{+}_{j,k+\frac{1}{2}}$, $a^{+}_{j+\frac{1}{2},k}$, respectivement dans la direction de y et x, sont les valeurs maximales et minimales des valeurs propres du Jacobien $\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{U}}$ et respectivement $\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{U}}$.

Un schéma est well-balanced si, lorsqu'on impose un état stationnaire comme une surface libre au repos, on retrouve une égalité entre le terme de pression et le terme de pente de fond dans l'équation de quantité de mouvement. Pour s'assurer de cette propriété, la discrétisation de la moyenne de la cellule pour le terme source a été proposée (Équation 3.15) : [17]

$$\overline{\boldsymbol{S}}_{j,k}(t) \approx \frac{1}{\Delta y \Delta x} \iint_{C_{j,k}} \boldsymbol{S}\big(\boldsymbol{U}(x,y,t), B(x,y)\big) dxdy$$

Équation 3.15

Les Équations 3.16 et 3.17 suivantes ont été proposées pour avoir une valeur non négative des composantes du terme moyen $\overline{\boldsymbol{S}}_{j,k}(t)$. [17]

$$\overline{\boldsymbol{S}}^{(2)}_{j,k}(t) \approx -g \frac{B_{j+\frac{1}{2},k} - B_{j-\frac{1}{2},k}}{\Delta x} . \frac{\left(w^{E}_{j,k} - B_{j+\frac{1}{2},k}\right) + \left(w^{W}_{j,k} - B_{j-\frac{1}{2},k}\right)}{2}$$

Équation 3.16

$$\overline{\boldsymbol{S}}^{(3)}_{j,k}(t) \approx -g \frac{B_{j,k+\frac{1}{2}} - B_{j,k-\frac{1}{2}}}{\Delta y} . \frac{\left(w^{N}_{j,k} - B_{j,k+\frac{1}{2}}\right) + \left(w^{S}_{j,k} - B_{j,k-\frac{1}{2}}\right)}{2}$$

Équation 3.17

Afin de s'assurer que la hauteur d'eau dans une cellule ne soit pas négative et que la fonction représentant l'élévation du fond puisse présenter des discontinuités, deux opérations ont été réalisées dans l'étape de discrétisation :

- Une approximation du fond par une fonction continue par morceaux
- Une reconstruction de w. [17]

Approximation du fond

L'élévation du fond $\tilde{B}(x,y)$ approximée sur une cellule $C_{j,k}$ est une fonction continue par morceaux et bilinéaire. Cette fonction est telle qu'avec la condition de continuité aux bords, la moyenne sur la surface de cellule de la fonction de reconstruction est égale à la valeur de celle-ci au centre des cellules et aussi égale à la moyenne des valeurs au milieu de chaque côté de la cellule de la fonction d'approximation. On a donc un remplacement de la fonction $B(x,y)$ continue par une approximation $\tilde{B}\left(x_j, y_k\right)$ de l'Équation 5.18. [17]

$$B_{j,k} \coloneqq \tilde{B}(x_j, y_k) = \frac{1}{\Delta y \Delta x} \iint_{C_{j,k}} \tilde{B}(x,y)\, dxdy = \frac{1}{4}\left(B_{j+\frac{1}{2},k} + B_{j-\frac{1}{2},k} + B_{j,k+\frac{1}{2}} + B_{j,k-\frac{1}{2}}\right)$$

Équation 3.18

Avec,

$$B_{j+\frac{1}{2},k} = \frac{1}{2}\left(B_{j+\frac{1}{2},k+\frac{1}{2}} + B_{j+\frac{1}{2},k-\frac{1}{2}}\right)$$

$$B_{j,k+\frac{1}{2}} = \frac{1}{2}\left(B_{j+\frac{1}{2},k+\frac{1}{2}} + B_{j-\frac{1}{2},k+\frac{1}{2}}\right)$$

Équation 3.19

Où $B_{j+\frac{1}{2},k+\frac{1}{2}}$ sont les valeurs de la fonction $\tilde{B}$ au coin de la cellule.

De cette dernière équation, on peut exprimer la valeur de l'élévation du fond au centre de la cellule avec l'Équation 3.20.

$$B_{j,k} = \frac{1}{2}\left(B_{j+\frac{1}{2},k} + B_{j-\frac{1}{2},k}\right) = \frac{1}{2}\left(B_{j,k+\frac{1}{2}} + B_{j,k-\frac{1}{2}}\right)$$
$$\overline{w}_{j,k} = \frac{w^E_{j,k} + w^W_{j,k}}{2} = \frac{w^S_{j,k} + w^N_{j,k}}{2}$$

Équation 3.20

Cette transformation a été intégrée dans l'expression des Équations 3.10, 3.16, 3.17.

Reconstruction de *w*

Cette procédure de reconstruction de *w* est réalisée via l'application des équations et étapes ci-dessous.

Une limitation de la dérivée de $\boldsymbol{U}$ est imposée pour limiter la pente de U au sein de la cellule par rapport aux cellules voisines, le limiteur est une fonction minmod exprimée à l'Équation 5.21 et 5.22. [17]

$$\left(\frac{\partial \boldsymbol{U}}{\partial x}\right)_{j,k} = minmod\left(\theta \frac{\overline{\boldsymbol{U}}_{j,k} - \overline{\boldsymbol{U}}_{j-1,k}}{\Delta x}, \frac{\overline{\boldsymbol{U}}_{j+1,k} - \overline{\boldsymbol{U}}_{j-1,k}}{2\Delta x}, \theta \frac{\overline{\boldsymbol{U}}_{j+1,k} - \overline{\boldsymbol{U}}_{j,k}}{\Delta x}\right)$$

Équation 3.21

$$\left(\frac{\partial \boldsymbol{U}}{\partial y}\right)_{j,k} = minmod\left(\theta \frac{\overline{\boldsymbol{U}}_{j,k} - \overline{\boldsymbol{U}}_{j,k-1}}{\Delta y}, \frac{\overline{\boldsymbol{U}}_{j,k+1} - \overline{\boldsymbol{U}}_{j,k-1}}{2\Delta y}, \theta \frac{\overline{\boldsymbol{U}}_{j,k+1} - \overline{\boldsymbol{U}}_{j,k}}{\Delta y}\right)$$

Équation 3.22

Le paramètre θ est un paramètre se référant à la viscosité numérique présente dans le schéma. Sa valeur doit se trouver dans l'intervalle $[1,2]$, une valeur importante correspond à un système moins dissipatif numériquement. Les valeurs de la hauteur d'eau sur les bords d'une cellule $h_{j,k}^{E,W,N,S}$ est obtenue par la formule suivante après l'approximation de l'élévation du fond comme une fonction continue par morceaux bi-linéaire : [17]

$$h_{j,k}^{E} \coloneqq w_{j,k}^{E} - B_{j+\frac{1}{2},k}\,,\; h_{j,k}^{W} \coloneqq w_{j,k}^{W} - B_{j-\frac{1}{2},k}$$

$$h_{j,k}^{N} \coloneqq w_{j,k}^{N} - B_{j,k+\frac{1}{2}}\,,\; h_{j,k}^{S} \coloneqq w_{j,k}^{S} - B_{j,k-\frac{1}{2}}$$

Équation 3.23

Pour s'assurer de la positivité de $h_{j,k}^{E,W,N,S}$, si la valeur est négative alors , il y a une imposition d'une hauteur nulle et une modification du bord opposé, selon les équations suivantes : [17]

$$\text{Si } w_{j,k}^{E} < B_{j+\frac{1}{2},k} \text{ alors } w_{j,k}^{E} = B_{j+\frac{1}{2},k}\,,\; w_{j,k}^{W} = 2\,\overline{w}_{j,k} - B_{j+\frac{1}{2},k} \text{ et}$$

$$\left(\frac{\partial w}{\partial x}\right)_{j,k} = \frac{B_{j+\frac{1}{2},k} - \overline{w}_{j,k}}{\Delta x/2}$$

Équation 3.24

$$\text{Si } w_{j,k}^{W} < B_{j-\frac{1}{2},k} \text{ alors } w_{j,k}^{E} = 2\,\overline{w}_{j,k} - B_{j-\frac{1}{2},k}\,,\; w_{j,k}^{W} = B_{j-\frac{1}{2},k} \text{ et}$$

$$\left(\frac{\partial w}{\partial x}\right)_{j,k} = \frac{\overline{w}_{j,k} - B_{j-\frac{1}{2},k}}{\Delta x/2}$$

Équation 3.25

$$\text{Si } w_{j,k}^{N} < B_{j,k+\frac{1}{2}} \text{ alors } w_{j,k}^{N} = B_{j,k+\frac{1}{2}}\,,\; w_{j,k}^{S} = 2\,\overline{w}_{j,k} - B_{j,k+\frac{1}{2}} \quad \text{et}$$

$$\left(\frac{\partial w}{\partial y}\right)_{j,k} = \frac{B_{j,k+\frac{1}{2}} - \overline{w}_{j,k}}{\Delta y/2}$$

Équation 3.26

$$\text{Si } w_{j,k}^{S} < B_{j,k-\frac{1}{2}} \text{ alors } w_{j,k}^{N} = 2\,\overline{w}_{j,k} - B_{j,k-\frac{1}{2}}\,,\; w_{j,k}^{S} = B_{j,k-\frac{1}{2}} \quad \text{et}$$

$$\left(\frac{\partial w}{\partial y}\right)_{j,k} = \frac{\overline{w}_{j,k} - B_{j,k-\frac{1}{2}}}{\Delta y/2}$$

Équation 3.27

Cette procédure garantit que la hauteur d'eau est non négative à la fois sur les bords et au centre de la cellule comme il a été démontré dans [17]. Cependant, la hauteur d'eau peut être nulle ou très faible, ce qui empêcherait de calculer les vitesses aux bords de manière précise. Une modification sur

le calcul des vitesses a donc été introduite, en remplaçant les vitesses par l'Équation 3.28 qui seront à introduire dans l'Équation 3.10.

$$u_{j,k} = \frac{\sqrt{2}\, h_{j,k}(u_{j,k}.h_{j,k})}{\sqrt{h^4{}_{j,k} + \max\left(h^4_{j,k}, \varepsilon\right)}} \text{ et } v_{j,k} = \frac{\sqrt{2}\, h_{j,k}(v.h_{j,k})}{\sqrt{h^4{}_{j,k} + \max\left(h^4_{j,k}, \varepsilon\right)}} \quad \text{Équation 3.28}$$

Avec ε une valeur faible choisie souvent égale à $\varepsilon = \max(\,(\Delta x)^4, (\Delta y)^4)$

Suite à cette dernière étape, les valeurs des débits spécifiques peuvent être recalculés sur base de cette vitesse :

$$(hu) \coloneqq h.u \text{ et } (hv) \coloneqq h.v \quad \text{Équation 3.29}$$

Les vitesses de propagation à travers un côté d'une cellule sont calculées avec les formules suivantes :

$$a^+_{j+\frac{1}{2},k} = max\left\{ u^E_{j,k} + \sqrt{gh^E_{j,k}},\ u^W_{j+1,k} + \sqrt{gh^W_{j+1,k}}, 0 \right\} \quad \text{Équation 3.30}$$

$$a^-_{j+\frac{1}{2},k} = max\left\{ u^E_{j,k} - \sqrt{gh^E_{j,k}},\ u^W_{j+1,k} - \sqrt{gh^W_{j+1,k}}, 0 \right\} \quad \text{Équation 3.31}$$

$$b^+_{j,k+\frac{1}{2}} = max\left\{ v^N_{j,k} + \sqrt{gh^N_{j,k}},\ v^S_{j,k+1} + \sqrt{gh^S_{j,k+1}}, 0 \right\} \quad \text{Équation 3.32}$$

$$b^-_{j,k+\frac{1}{2}} = max\left\{ v^N_{j,k} - \sqrt{gh^N_{j,k}},\ v^S_{j,k+1} - \sqrt{gh^S_{j,k+1}}, 0 \right\} \quad \text{Équation 3.33}$$

3.2.b Schéma d'intégration temporelle

Le schéma d'intégration temporelle est basé sur un algorithme explicite Runge-Kutta à deux pas.

On peut écrire l'Équation 3.8 sous la forme suivante :

$$\frac{d}{dt}\overline{\boldsymbol{U}}_{j,k}(t) = RHS_{j,k}$$

Avec $RHS_{j,k}$ le terme de droite de l'Équation 3.8, il peut être évalué par la procédure décrite Chapitre 3 -3.2.b . L'algorithme Runge-Kutta à deux pas, s'effectue à l'aide du calcul d'un prédicteur, Équation 3.34 et, d'un correcteur, Équation 3.35.

$$f^*_{j,k} = f^t_{j,k} + \Delta t.\, RHS^t_{j,k} \quad \text{Équation 3.34}$$

$$f_{j,k}^{t+\Delta t} = f_{j,k}^{t} + \Delta t.\left[(1-a_1)RHS_{j,k}^{t} + a_1\, RHS_{j,k}^{*}\right]$$

Équation 3.35

La valeur de la variable a_1 est égale à 0,5 car nous sommes dans le cas d'un algorithme Runge-Kutta de type RK22. Cette valeur permet alors une précision maximale et est largement utilisée dans le cas de calcul instationnaire.

Le terme pas de temps Δt sera défini de manière à assurer une stabilité du schéma. Le critère utilisé est le critère de Courant-Friedrich-Levy qui est une condition nécessaire et non suffisante de stabilité. Il est vérifié lorsque le domaine de dépendance numérique contient tout le domaine de dépendance physique, sous forme mathématique ce critère s'écrit avec l'Équation 3.36.

$$\frac{\Delta t}{\Delta x} = \frac{CFL}{|u| + c}$$

Équation 3.36

Avec Δt, le pas de temps, u la vitesse, c la celérité soit $\sqrt{g\,h}$ et CFL, le nombre de Courant. Pour satisfaire le critère, cette dernière variable doit avoir une valeur inférieure à 1.

3.3 Implémentation du schéma

Le schéma de résolution des Équations 3.8, a été implémenté et résolu dans le programme SARndbox sous la forme d'image et non de matrices avec l'utilisation des GSLS shaders pour les calculs d'écoulement.

Comme expliqué au Chapitre 3 -1.2 , un shader traite non pas une matrice mais une texture qui peut être vue comme une image, avec les données des pixels équivalentes aux valeurs des éléments de la matrices. Après les calculs effectués, les textures vont pouvoir être représentées à l'écran.

Dans le programme SARnbdox, les deux principales textures concerneront la bathymétrie et les inconnues liées à l'écoulement :

- `bathymetryTextureObject[2]` : Une texture de type GL_LUMINANCE contenant une composante égale à l'élévation du fond ($B_{j,k}$) sous forme de `float` . Les données sont des données centrées sur les sommets de la cellule. Elle est exprimée sous forme d'une texture de double-buffer. Le premier buffer pour effectuer les opérations de calcul et le second pour la représentation.
- `quantityTextureObject[3]` : Une texture de type GL_RGB contenant trois composante du vecteur des inconnues soit $\boldsymbol{U} = (w, hu, hw)$. La composante rouge égale à l'élévation de la surface libre (w), la composante verte correspondant au débit spécifique dans la direction de x (hu) et la composante bleue correspondant au débit spécifique dans la direction de y (hv). Les données sont des données centrées sur le centre de cellules. Sous

forme d'une texture de triple-buffer. Le premier est le résultat au temps précèdent, le second permet d'effectuer les opérations et le dernier pour la représentation.[3]

Ces deux textures seront sauvegardées à la suite de la simulation d'un pas de temps. D'autres textures temporaires propre à la simulation d'écoulement seront créées et effacées à la fin du pas calculé. Un tableau des textures et frame buffer liés est présenté au Tableau 7.

A ces textures vont être liés des frame buffer object permettant de stocker sur le GPU des résultats des opérations effectuées sur les composantes de textures. Une opération de transfert vers le CPU sera nécessaire pour extraire ces informations, il est utile de noter que cette opération est lente et donc à limiter.

Le Tableau 6 présente les shaders utilisés dans le code SARndbox pour la simulation d'écoulement ainsi que le fichier dans lequel ils sont appelés, mais aussi, s'il existe un fragment shader et un vertex shader propre au shader.

Shader	.vs	.fs	Utilisation (fonction, fichier)
Water2BathymetryUpdateShader		X	Fonction : updateBathymetry(), WaterTable2.cpp
Water2BoundaryShader		X	Fonction :runSimulationStep(), WaterTable2.cpp
Water2EulerStepShader		X	Fonction :runSimulationStep(), WaterTable2.cpp
Water2MaxStepSizeShader		X	Fonction :calDerivative(), WaterTable2.cpp
Water2RungeKuttaStepShader		X	Fonction :runSimulationStep(), WaterTable2.cpp)
Water2SlopeAndFluxAndDerivativeShader		X	Fonction :calDerivative(), WaterTable2.cpp
Water2WaterAddShader	X	X	Fonction :runSimulationStep (), WaterTable2.cpp
Water2WaterUpdateShader		X	Fonction :runSimulationStep(), WaterTable2.cpp

Tableau 6 : Liste des shaders utilisés pour la simulation d'écoulement

L'analyse se divisera en deux parties :

1. La boucle principale d'appel de simulation dans le fichier Sandbox.cpp
2. Les 2 fonctions du fichier `WaterTable2.cpp` appelées dans la boucle principale :

    a. `updateBathymetry()`

    b. `runSimulationStep()`

[3] Dans la suite de ce travail, le terme « texture quantité » sera utilisé pour signifier la `quantityTextureObject`

La légende de tous les schémas présentés dans cette section est disponible à la Figure 32.

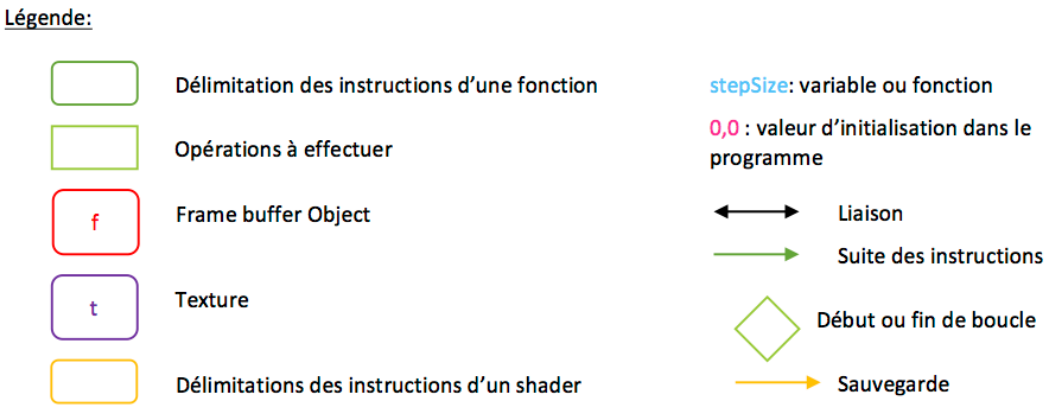


Figure 32: Légende des schémas blocs

3.3.a La boucle principale

La boucle principale est appelée non pas dans le main, fonction principale décrite au Chapitre 3 -2.1 , mais dans la fonction `display()` OpenGL appelée donc automatiquement.

La structure de cette fonction est disponible à la Figure 33.[4]

La boucle de simulation est exécutée pour chaque frame obtenu de la Kinect ce qui représente le temps simulé. A la suite de la boucle de simulation, le reste de la fonction `display()` est exécutée. Une fois arrivée à la fin de la fonction, elle est re-executée de suite. Le rendu visuel est donc mis à jour à la fin de chaque exécution de `display(.)`Pour un rendu visuel progressif, le nombre de pas effectués lors d'une simulation est limité à 30 pas pour avoir un rendu visuel à chaque fois que la Kinect capte un frame.

[4] Seules les parties du code qui concernent la simulation d'écoulement seront présentées dans le détails

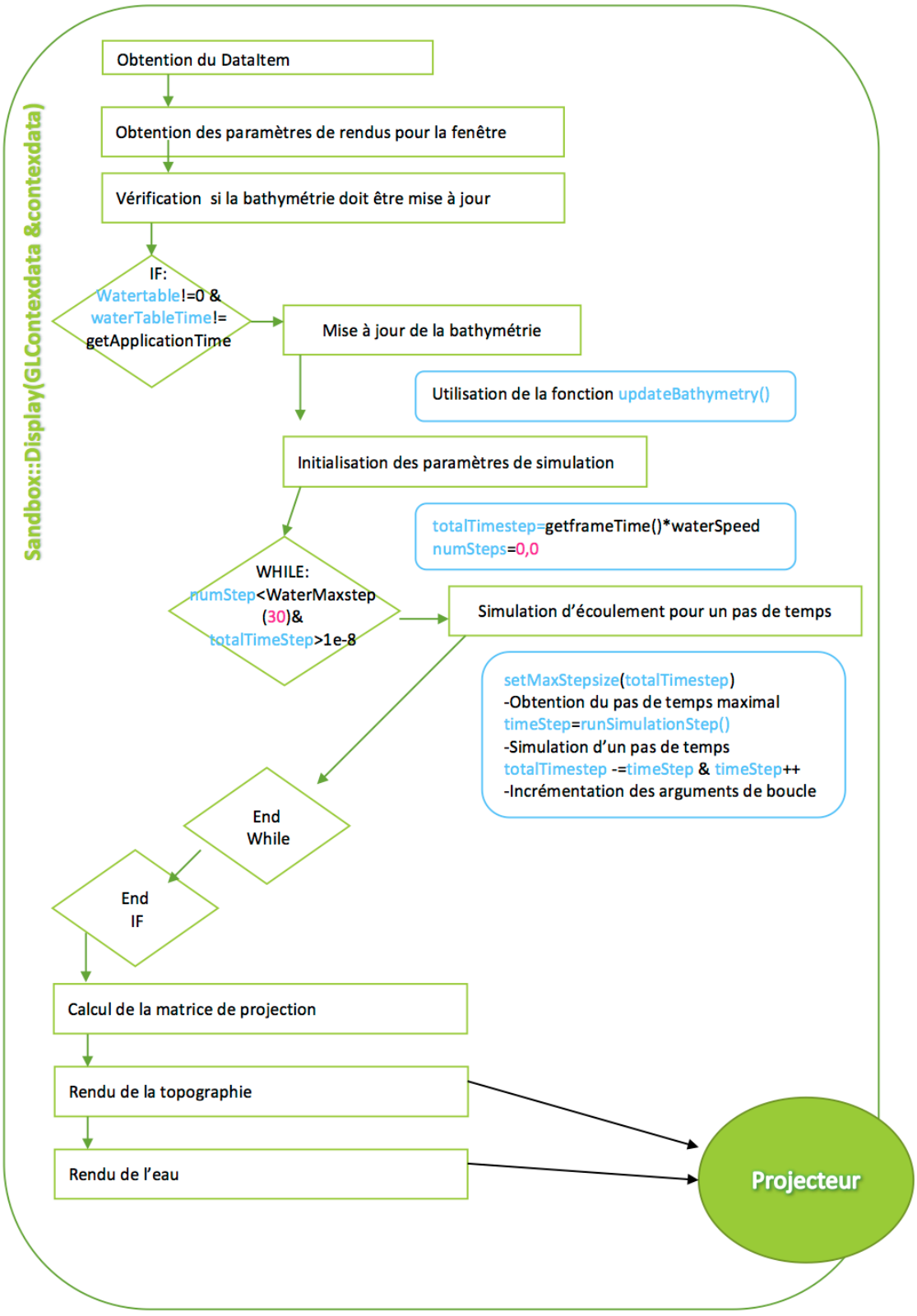


Figure 33: Schéma bloc de la boucle principale du programme SARndbox

3.3.b La fonction updateBathymetry()

Dans cette fonction, on associe le frame capté par la Kinect à la texture de bathymétrie. Sur base des valeurs de la texture de bathymétrie, on modifie alors les valeurs de la texture des quantités relatives à l'écoulement. Ces 2 textures seront identifiées comme les conditions initiales du schéma de résolution numérique. La structure de cette fonction est disponible à la Figure 34

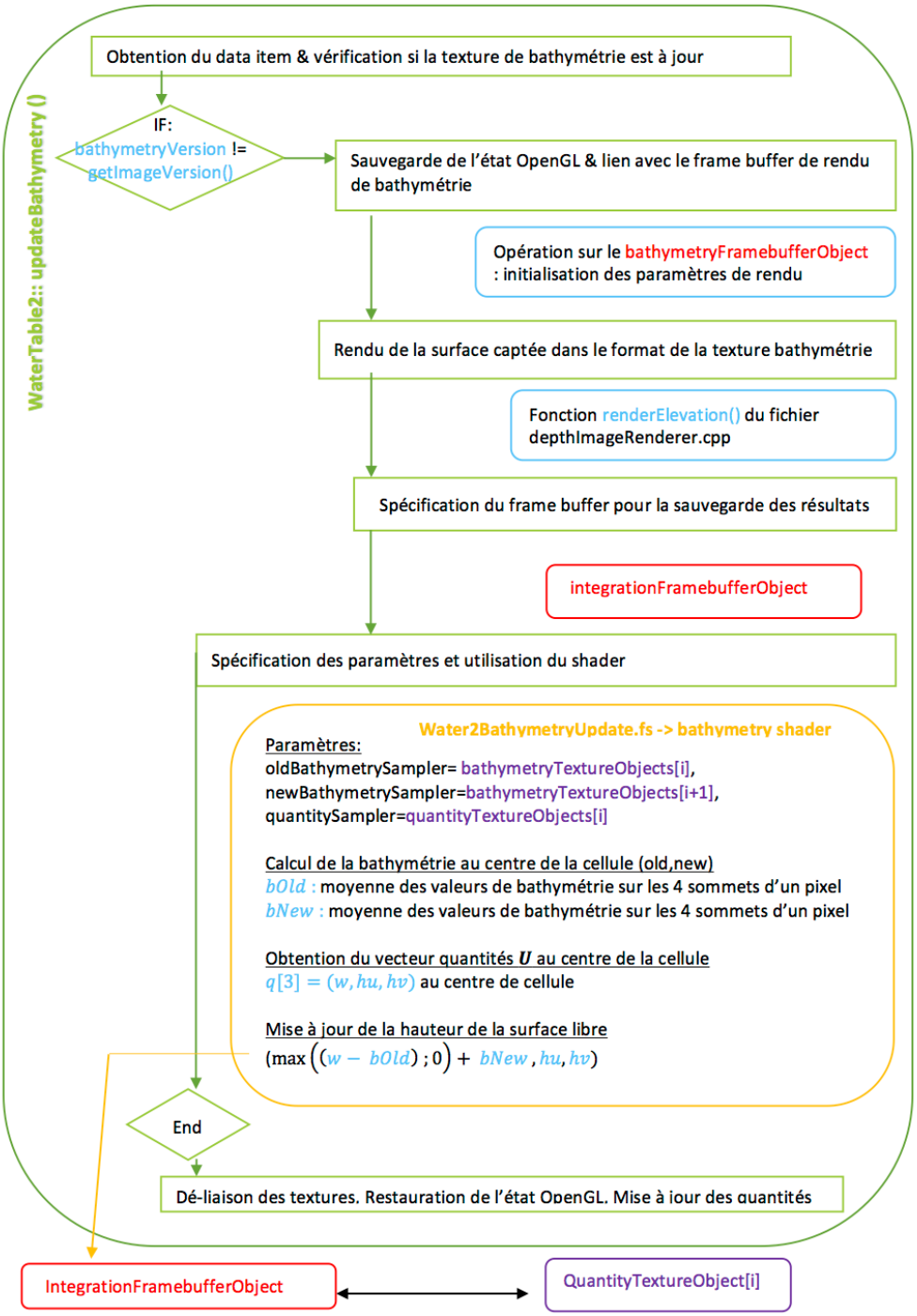


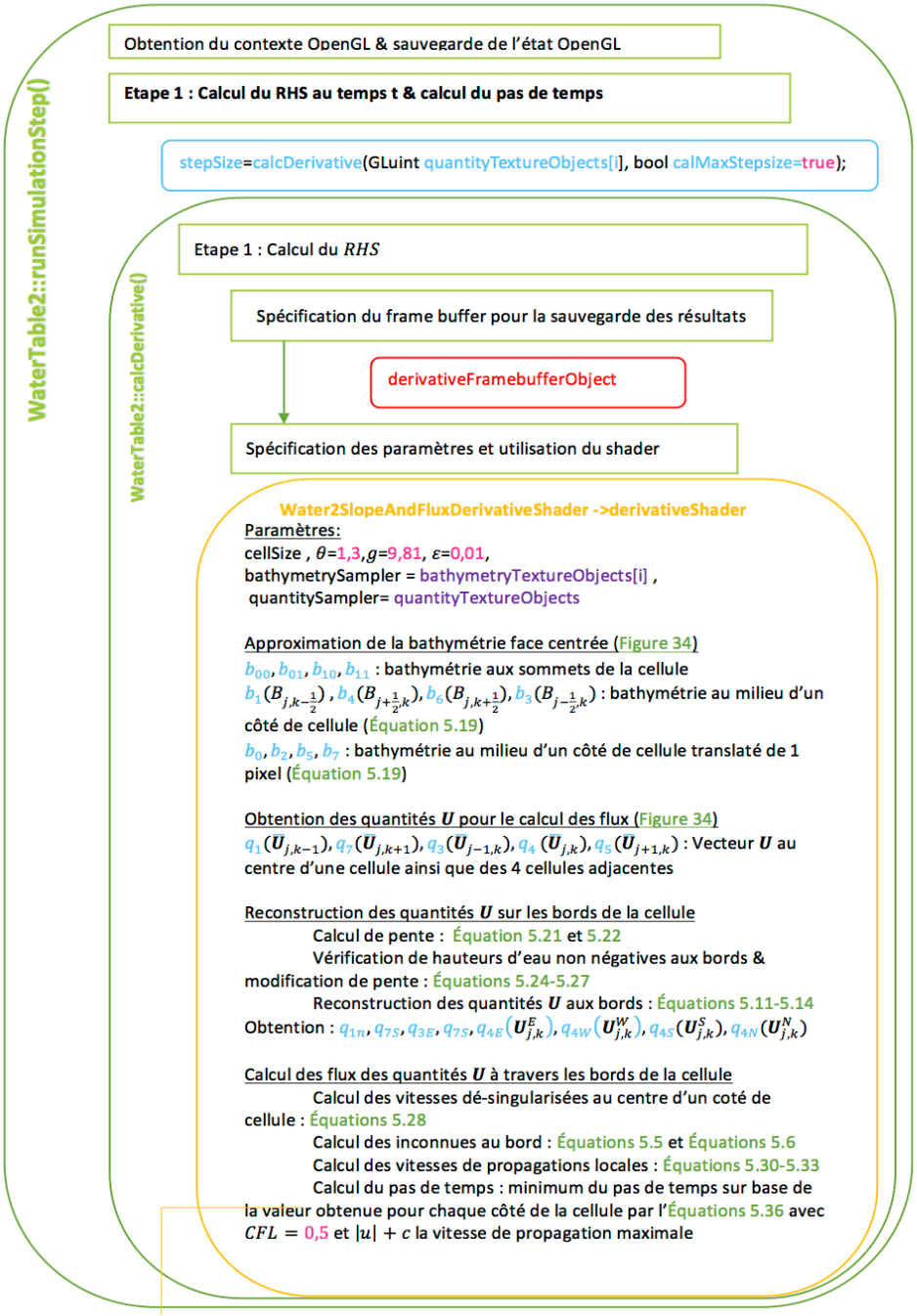
Figure 34: Schéma bloc de la fonction updateBathymetry()

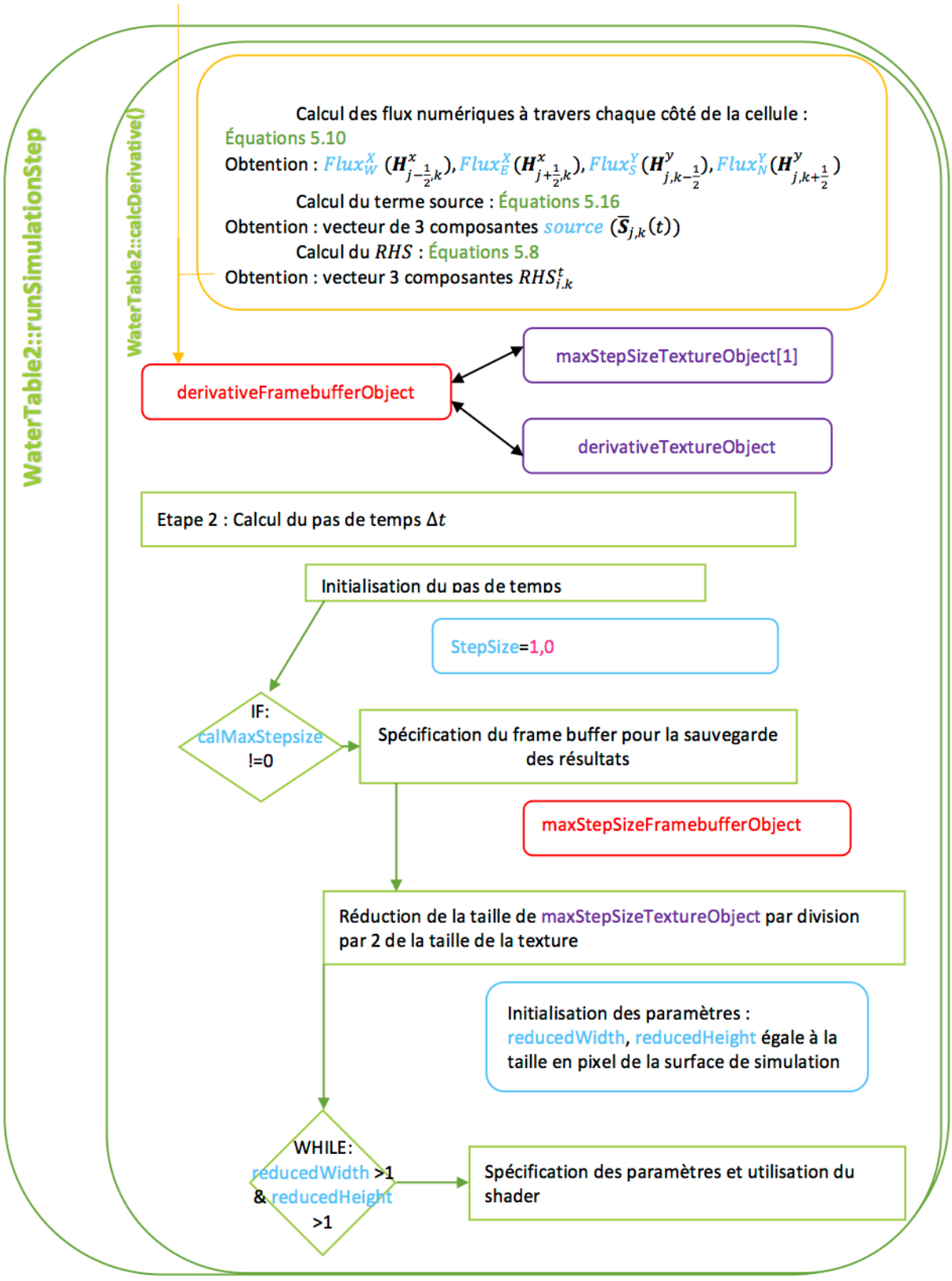
3.3.c La fonction runSimulationStep()

La structure de la fonction de simulation sera divisée en 6 étapes explicitées dans le code informatique afin de commenter ces étapes au fur et à mesure.

Etape 1 : Calcul du RHS au temps t

Le schéma de l'étape 1 de la fonction `runSimulationStep()` est disponible à la Figure 35.




WaterTable2::runSimulationStep
WaterTable2::calcDerivative()
Calcul des flux numériques à travers chaque côté de la cellule :
Équations 5.10
Obtention : $Flux_W^X(\boldsymbol{H}^x_{j-\frac{1}{2},k}), Flux_E^X(\boldsymbol{H}^x_{j+\frac{1}{2},k}), Flux_S^Y(\boldsymbol{H}^y_{j,k-\frac{1}{2}}), Flux_N^Y(\boldsymbol{H}^y_{j,k+\frac{1}{2}})$
Calcul du terme source : Équations 5.16
Obtention : vecteur de 3 composantes $source\ (\overline{\boldsymbol{S}}_{j,k}(t))$
Calcul du RHS : Équations 5.8
Obtention : vecteur 3 composantes $RHS^t_{i.k}$
derivativeFramebufferObject
maxStepSizeTextureObject[1]
derivativeTextureObject
Etape 2 : Calcul du pas de temps Δt
Initialisation du pas de temps
StepSize=1,0
IF: calMaxStepsize !=0
Spécification du frame buffer pour la sauvegarde des résultats
maxStepSizeFramebufferObject
Réduction de la taille de maxStepSizeTextureObject par division par 2 de la taille de la texture
Initialisation des paramètres : reducedWidth, reducedHeight égale à la taille en pixel de la surface de simulation
WHILE: reducedWidth >1 & reducedHeight >1
Spécification des paramètres et utilisation du shader

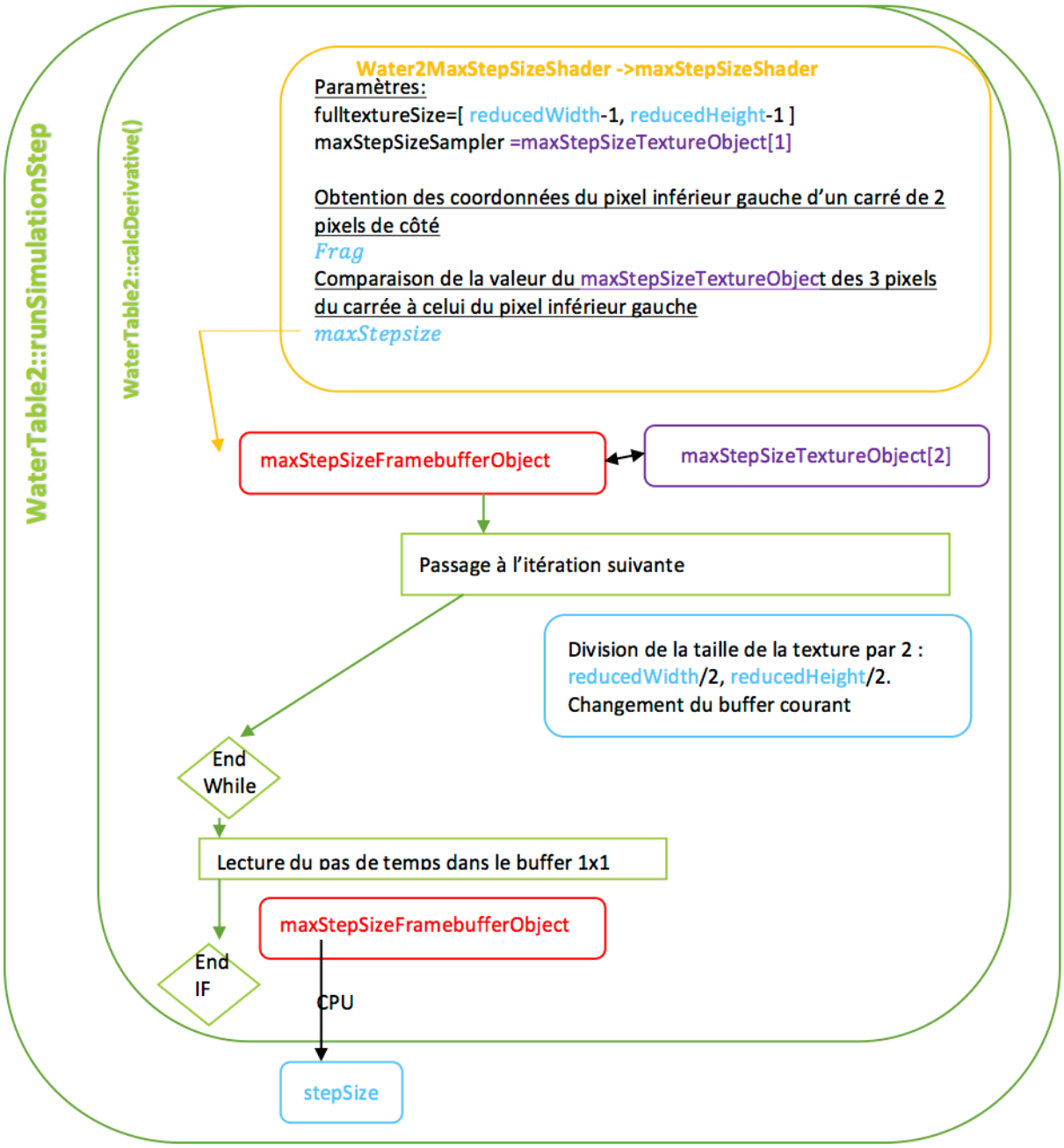


Figure 35: Schéma bloc de l'étape 1 de la fonction runSimulationStep()

Cette première étape permet de calculer le $RHS_{j,k}^{t}$ et le pas de temps Δt afin de calculer le premier pas du schéma d'intégration temporelle. Le schéma volume fini pour la discrétisation spatiale présenté au Chapitre 3 -3.2.a a été implémenté avec les paramètres $\varepsilon = 0{,}01$ et $\theta = 1{,}3$.

Le pas de temps calculé basé sur un critère de Courant a été implémenté de manière à trouver le pas de temps minimum en comparant les pixels de toute la texture. La valeur du nombre de Courant a été imposé à 0,5. Il est utile de noter que seule la variable stepSize sera retournée explicitement au

CPU comme résultat de la fonction `calDerivative()`. Les textures modifiées sont retournées via le buffer auxquelles elles sont liées. On remarquera aussi qu'aucun terme de frottement n'a été pris en compte dans le calcul du terme source.

La Figure 36 présente la disposition de variables calculées dans le shader derivativeShader au sein d'une cellule.

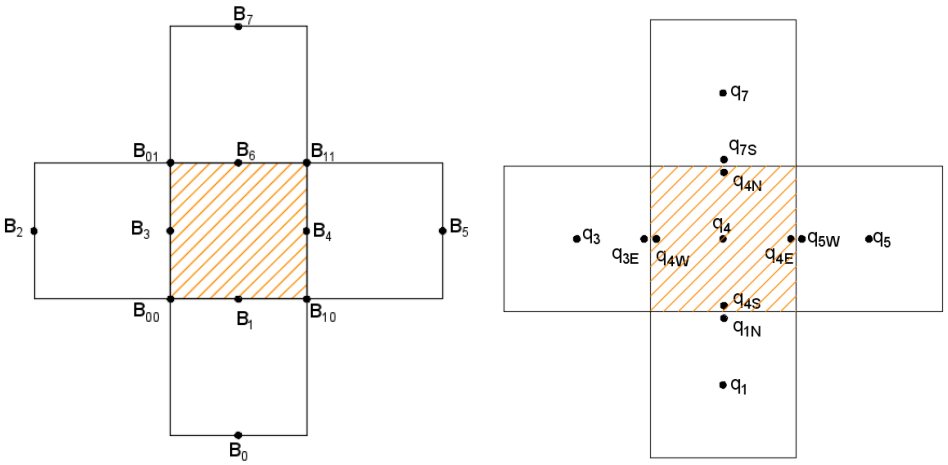


Figure 36: Disposition des variables utilisées dans le shader Water2SlopeAndFluxAndDerivativeShader: des inconnues (à droite) et de la bathymétrie (à gauche), hachures= cellule de travail

Etape 2 : Calcul du prédicteur

La Figure 37 présente le schéma bloc de la seconde étape dans la fonction `runSimulationStep()`.

Dans cette étape du calcul du premier pas du schéma d'intégration temporelle, une particularité apparaît. En effet, après le calcul du prédicteur, les débits spécifiques dans les deux directions sont diminués de plus ou moins 1%. Aucune justification dans la présentation du modèle de discrétisation spatiale ou explication physique n'a été trouvée, cela pourrait être une modification pour prendre en compte un frottement de manière numérique.

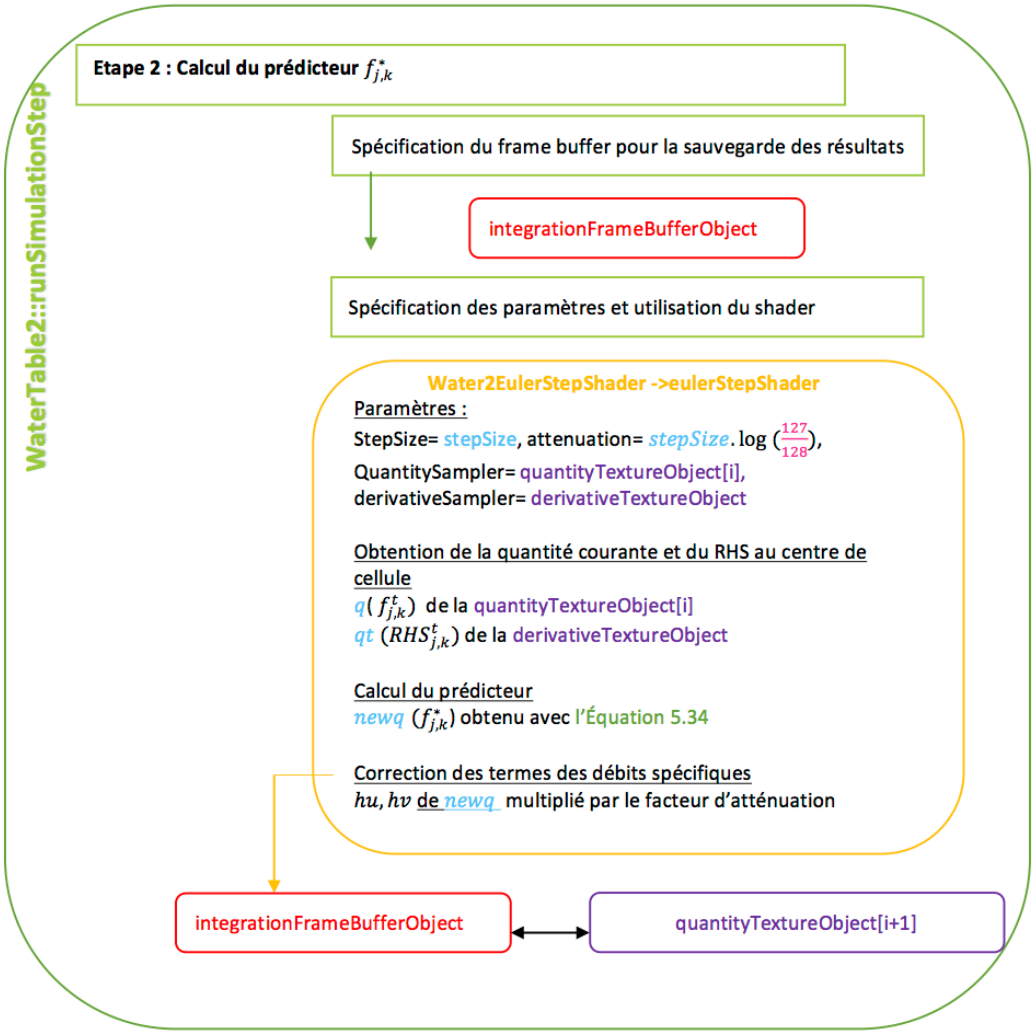


Figure 37 : Schéma bloc de l'étape 2 de la fonction runSimulationStep()

Etape 3 : Calcul du $RHS^*_{j,k}$

La Figure 38 présente le schéma bloc de la troisième étape dans la fonction `runSimulationStep()`.

Pour le calcul du $RHS^*_{j,k}$, la fonction `calDerivative()` utilisée pour l'étape 1, est aussi utilisée dans cette étape mais seulement la première partie de cette fonction puisque le pas de temps a déjà été déterminé lors de l'étape 1.

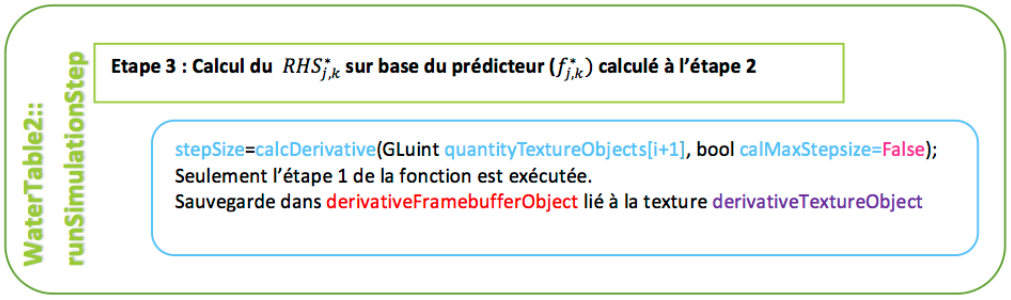
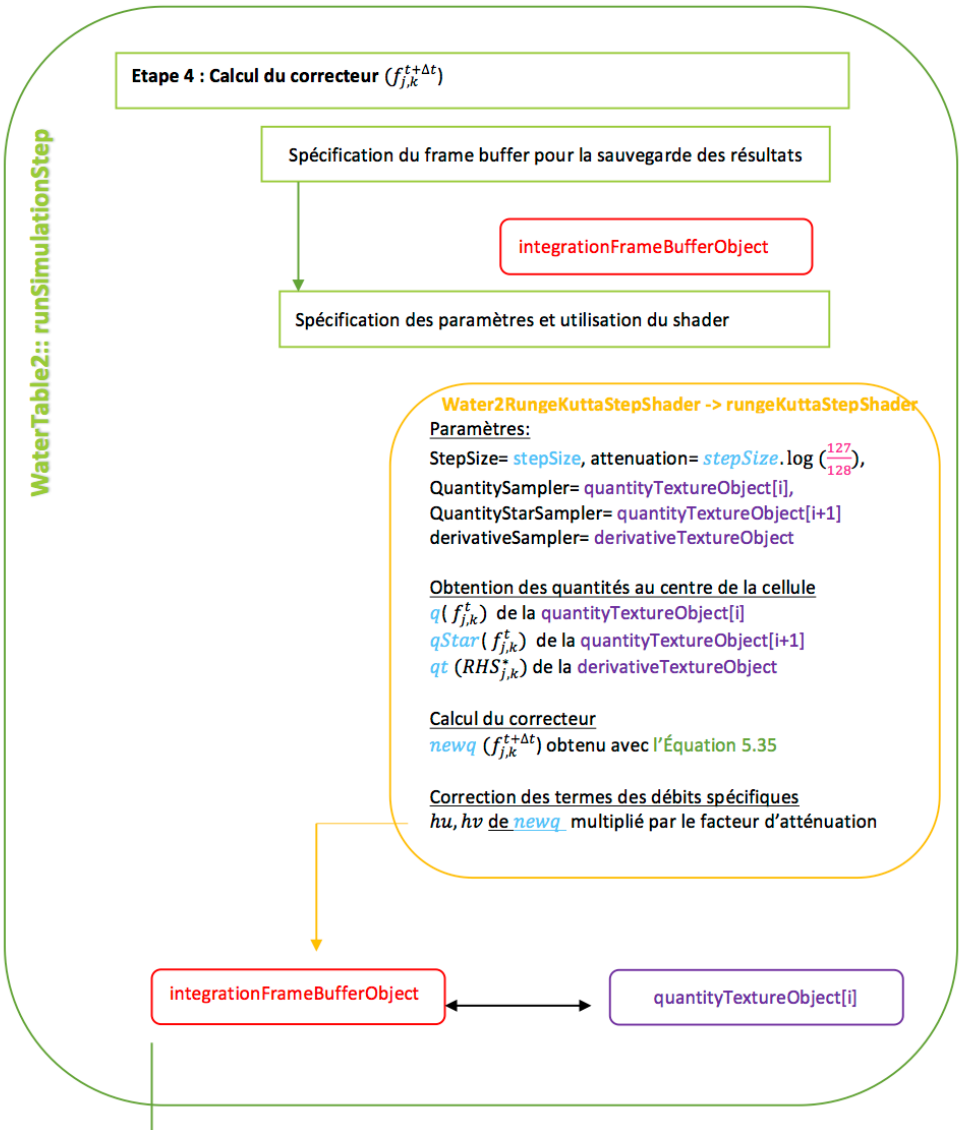


Figure 38: Schéma bloc de l'étape 3 de la fonction runSimulationStep()

Etape 4 : Calcul du correcteur et imposition des conditions aux limites

La Figure 39 présente le schéma bloc de la quatrième étape dans la fonction `runSimulationStep()`.



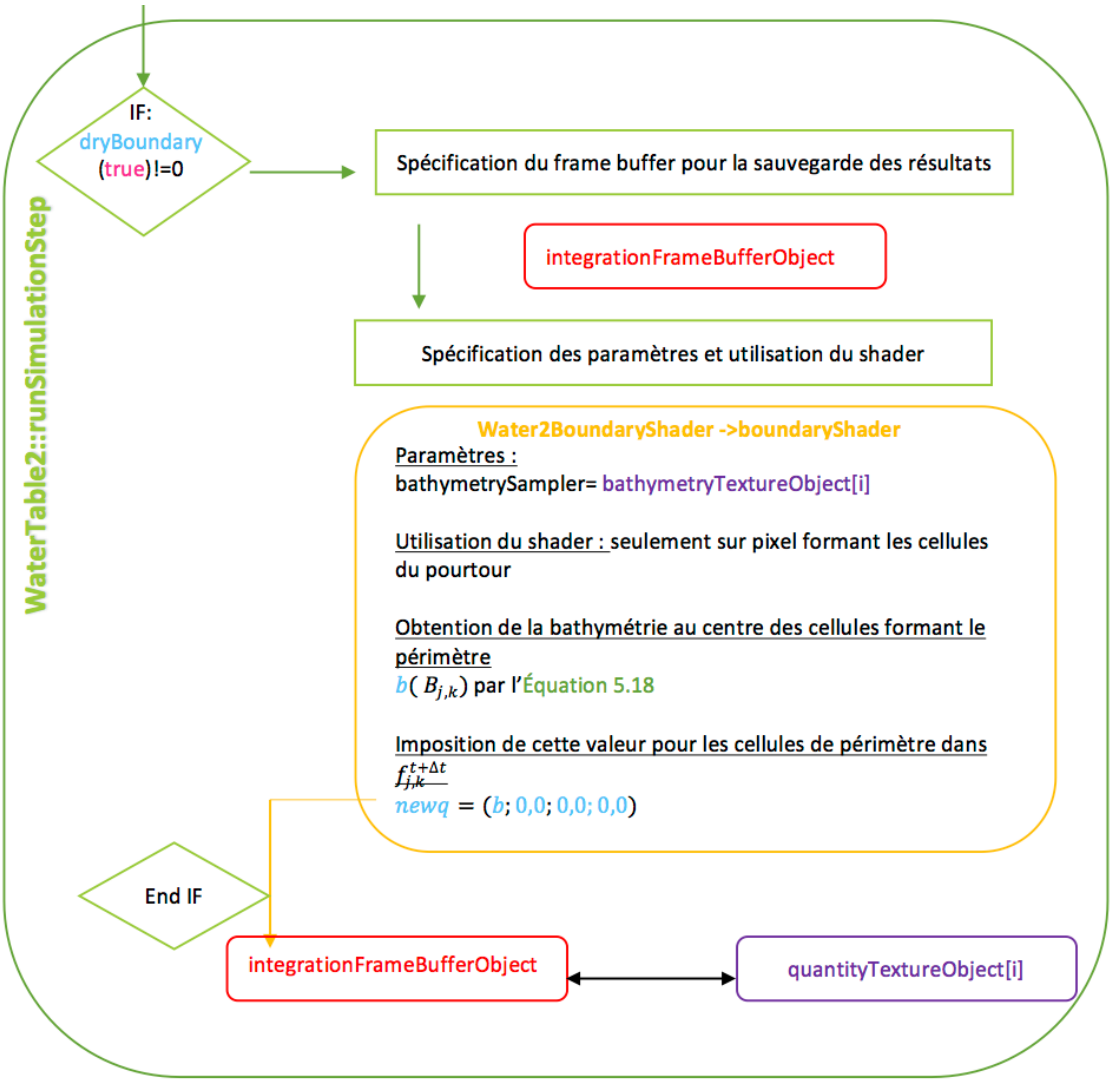


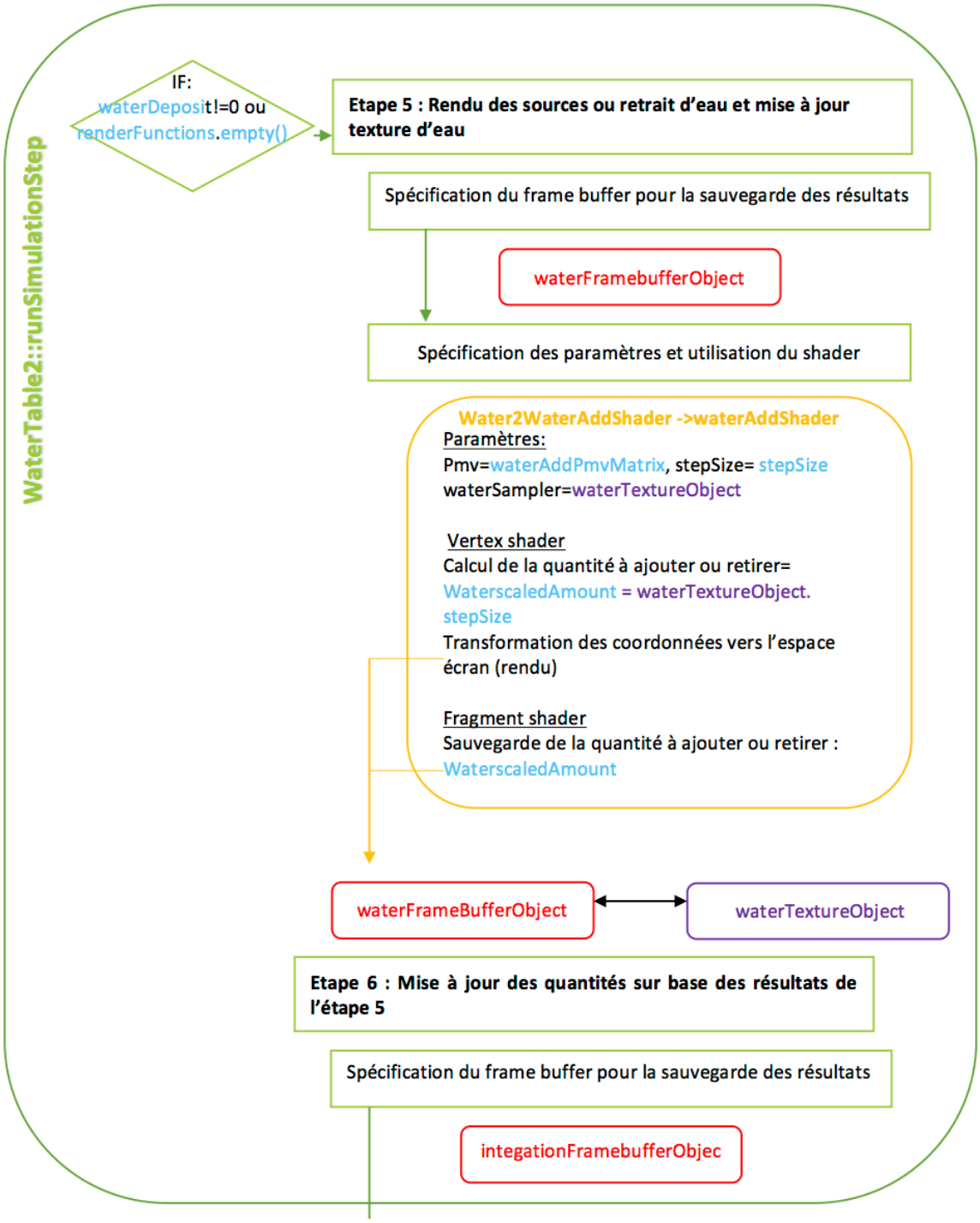
Figure 39: Schéma bloc de l'étape 4 de la fonction runSimulationStep()

Dans cette quatrième étape, le correcteur est calculé, on obtient donc les quantités (w, hu, hv) au temps $t + \Delta t$. Dans cette étape comme à l'étape 2, il y a une réduction forfaitaire des débits spécifiques sur les quantités finales de plus ou moins 1%.

La deuxième opération est l'imposition des conditions aux limites. Dans le programme, la condition aux limites des inconnues (w, hu, hv) est une condition forte en imposant une hauteur d'eau nulle dans les pixels du périmètre de la texture quantité en égalant la variable d'élévation de la surface libre (w) égale à la topographie reconstruite au centre du pixel. En conséquence, l'eau qui atteint un bord du domaine est enlevée du modèle, la masse d'eau sur le modèle n'est donc pas conservée. On impose aussi les conditions aux limites de débit spécifique nul aux bords.

Etape 5 et 6 : Mise à jour des textures

La Figure 40 présente le schéma bloc de la cinquième et sixième étapes dans la fonction `runSimulationStep()`.


WaterTable2::runSimulationStep
IF:
waterDeposit!=0 ou
renderFunctions.empty()
Etape 5 : Rendu des sources ou retrait d'eau et mise à jour texture d'eau
Spécification du frame buffer pour la sauvegarde des résultats
waterFramebufferObject
Spécification des paramètres et utilisation du shader
Water2WaterAddShader ->waterAddShader
Paramètres:
Pmv=waterAddPmvMatrix, stepSize= stepSize
waterSampler=waterTextureObject
Vertex shader
Calcul de la quantité à ajouter ou retirer=
WaterscaledAmount = waterTextureObject.
stepSize
Transformation des coordonnées vers l'espace écran (rendu)
Fragment shader
Sauvegarde de la quantité à ajouter ou retirer :
WaterscaledAmount
waterFrameBufferObject
waterTextureObject
Etape 6 : Mise à jour des quantités sur base des résultats de l'étape 5
Spécification du frame buffer pour la sauvegarde des résultats
integationFramebufferObjec

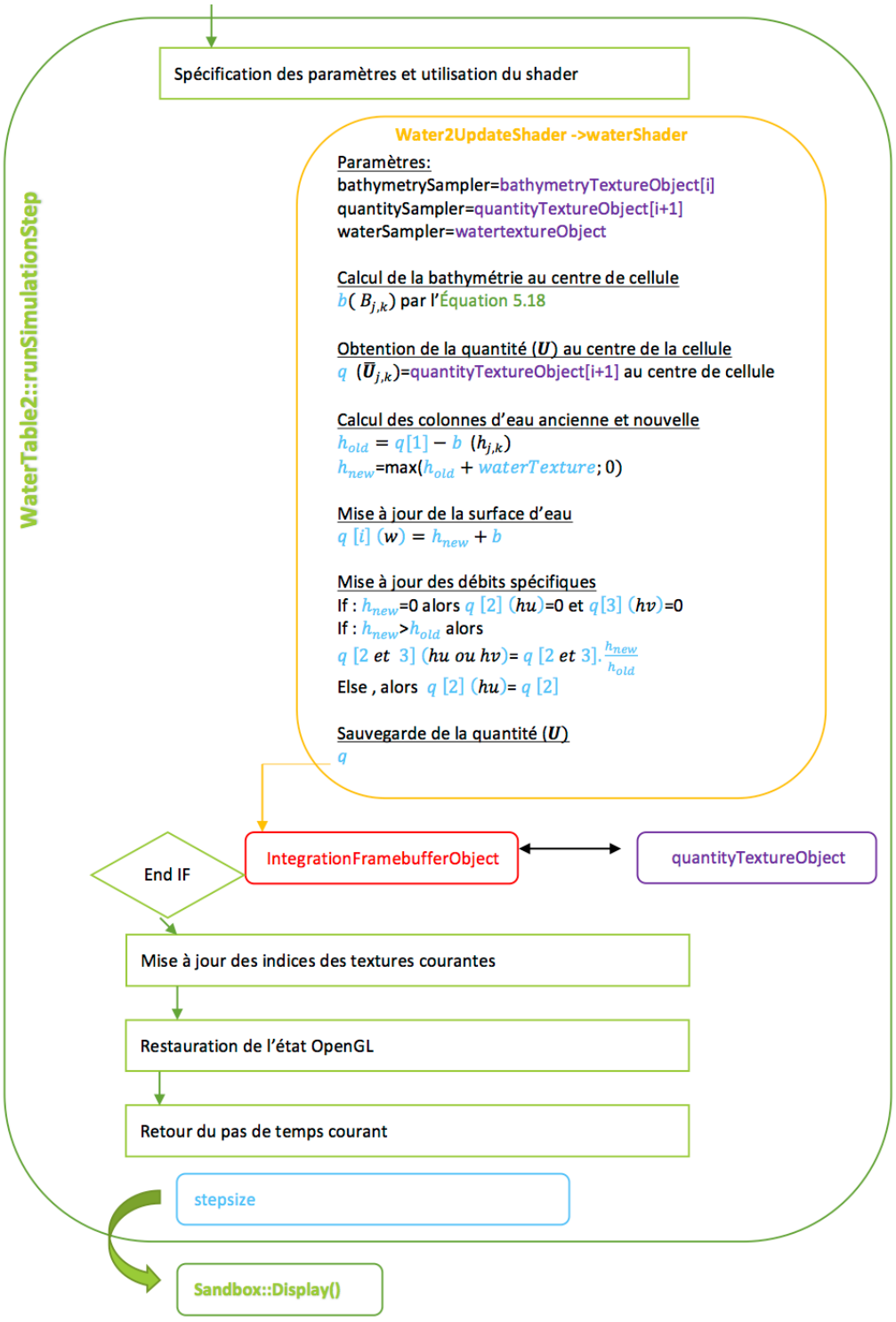


Figure 40: Schéma bloc des étape 5 et 6 de la fonction runSimulationStep()

La première partie de cette opération est l'ajout d'eau forfaitaire ou le retrait en fin de pas de temps. En effet, une option d'évaporation ou de condensation peut être activée à l'exécution du

programme, en imposant une quantité d'eau à ajouter ou retirer à la fin de la boucle de simulation. La création de cette texture qui possède les caractéristiques de rendu, passe par l'utilisation d'une matrice de projection et de modèle. Cet objet OpenGL permet de passer du repère interne à une objet OpenGL vers un repère propre à l'application fenêtrée. Cette transformation est nécessaire pour permettre une superposition des différentes textures lorsqu'elles sont représentées.

La seconde partie de cette étape, est la mise à jour du vecteur des quantités (w, hu, hv) au temps $t + \Delta t$ sur base d'un ajout ou retrait. L'élévation de la surface libre (w) est modifiée en conséquences. Quant au débit spécifique, la modification de celui-ci est telle que l'ajout d'eau se fait à vitesse nulle et que le retrait se fait à vitesse constante.

Afin de résumer toutes les textures et frame buffer qui ont été créés de manière temporaire ou permanentes, le Tableau 7 reprend la liste de ceux-ci et leurs caractéristiques principales.

Frame buffer object	Texture liée	Temporaire (T) ou Permanente (P)	Cellule centrée (CC) ou centré sur les sommets (CS)
integrationFramebufferObject	quantityTextureObjects	P	CC
derivativeFramebufferObject	quantityTextureObjects	P	CC
	maxStepSizeTextureObjects	T	CC
maxStepSizeFramebufferObject	maxStepSizeTextureObjects	T	CC
waterFramebufferObject	waterTextureObject	T	CC
BathymetryFrameBufferObject	bathymetryTextureObject	T	CS

Tableau 7 : Récapitulatif des textures, frame buffer liés et des caractéristiques

3.3.d Représentation des nombres

La représentation des nombres est importante pour connaitre la précision sur les valeurs obtenues.

Dans le programme SARndbox, les valeurs captées et calculées sont de type float avec une représentation en virgule flottante de type simple précision. Un chiffre réel peut être écrit en virgule flottante à partir d'un matisse et d'un exposant, de manière à ce que l'expression suivante soit exact :

$$nombre_{reel} = mantisse \, . \, e^{exposant}$$

Dans la représentation en virgule flottante simple précision, le nombre encodé sur 4 octets soit 32 bits avec 1 bit de signe, 8 bits d'exposants et 23 bits de mantisse. Cette représentation assure 7 chiffres significatifs du nombre réel calculé ou défini.

3.3.e Conditions aux limites et conditions initiales

Le nombre de conditions initiales peut être obtenu de la pente des caractéristiques de l'écoulement. Les deux vitesses relatives de propagation à travers les bords d'une cellule dans la direction x et dans la direction y sont de signes opposés, cela nous indique que nous sommes en présence d'un flux sub-critique aussi expliqué par une pente de fond faible. A parti de ces indications on peut donc déterminer le nombre de conditions initiales et aux limites.

Deux conditions initiales doivent être spécifiées. Les conditions sont donc une hauteur d'eau initiale dans tout le domaine égale à zéro et des débits spécifiques nuls dans les directions x et y.

Les conditions aux limites sont imposées dans le cas du programme SARndbox après la boucle de calcul d'un pas de temps en retirant, sur toutes les cellules du périmètre, l'eau présente qui a été calculée et en annulant les débits spécifiques. Normalement ces conditions fortes auraient dû être imposées au début de la boucle de simulation et seulement la valeur d'une inconnue aurait dû être imposée. Des plus, dans chaque direction, il aurait fallu une condition amont et une condition aval puisqu'il y a deux caractéristiques entrantes dans chaque direction.

Chapitre 4 Adaptation à la Kinect V2

Après avoir analysé les composantes d'un bac à sable à réalité augmentée ainsi que la structure du programme, ce chapitre aborde la partie du travail consacré à la modification du programme SARndbox en vue de l'utilisation d'une Kinect v2 à la place d'une Kinect v1 actuellement supportée. Ce chapitre présentera les étapes réalisées.

La captation d'un relief par la Kinect sous forme d'un frame de profondeur est gérée par des appels de fonctions du programme Kinect 3D Video Processing dans le programme SARndbox. Suivant les spécifications du développeur, la version 3.2 de Kinect 3D Video Processing est une version supportant expérimentalement la Kinect XBox One soit la seconde génération de Kinect.

1. Interaction avec la Kinect v2

La première étape du cheminement a été de vérifier que la Kinect v2 est détectée sur l'ordinateur et dès lors qu'il est possible d'interagir avec l'appareil.

Les essais ont été menés pour obtenir les paramètres internes de la Kinect v2 en exécutant l'exécutable `KinectUtil` propre au programme Kinect 3D Video Processing. Cet utilitaire permet de lister, détecter et configurer les appareils. L'option `list` lors de l'exécution permet de détecter la Kinect connectée. Á l'exécution de cette méthode, le programme retournait des messages d'erreurs spécifiant qu'aucun appareil compatible n'était détecté alors que les couleurs des diodes de la Kinect correspondaient bien à un dispositif allumé et que la Kinect apparaissait dans les dispositifs connectés à l'ordinateur (port USB 3). L'analyse du code et de l'erreur était la conséquence de non détection du numéro de série et type de caméra via les connexions USB de l'ordinateur.

Au vu de cette erreur et après avoir essayé de contourner cette première détection, alors que la compatibilité du programme à la Kinect v2 était censée être assurée, le choix a été fait d'utiliser une autre librairie Open source fonctionnant avec certitude avec la Kinect v2 dans l'environnement LINUX. Notre choix s'est porté sur une librairie OpenSource Libfreenect2. En effet, pour l'environnement de développement sous UNIX les librairies du fournisseur Microsoft ne sont pas disponibles, de plus la librairie choisie est très bien structurée bien que très différente de la librairie utilisée dans le programme SARnbox..

1.1 La librairie Lifreenect2

La librairie libfreenect2 est une libraire Open Source permettant des accès à la Kinect v2, elle est composée d'une librairie dynamique pour utilisation dans des programmes home made et d'applications exemple déjà développées.

Après l'installation de ce programme, de son environnement de développement sur l'ordinateur et la génération d'un projet dans l'environnement de développement, le programme exemple

`Protonect` a été exécuté. Grâce aux essais de diverses configurations et des messages d'erreur plus explicites de cette librairie, une configuration a permis la détection de la Kinect V2. La seule configuration qui fonctionne est celle où la Kinect est connectée sur le port USB 3.0 directement connecté sur la carte mère. Cela n'est pas documenté et est probablement lié soit à la puissance d'alimentation de ce port USB particulier ou à l'accès direct des drivers bas niveau au port matériel.

L'exécution du programme `Protonect` avec une Kinect placée parallèlement à un mur blanc à une distance de plus ou moins 1,4m a produit le résultat à l'écran présenté à la Figure 41. Un autre exemple du rendu visuel est disponible à la Figure 42 qui correspond à la même configuration avec un caisse (32 x 28 x 21cm) placée dans le champ visuel.

Cette application fenêtrée présente 4 vidéos sur un même écran :

- en bas à gauche, la vidéo présente l'image RGB de la caméra couleur
- en bas à droite, l'image de distance de la caméra infrarouge
- en haut à droite, une image de superposition des images RGB et infrarouge
- en haut à gauche, l'image infrarouge.

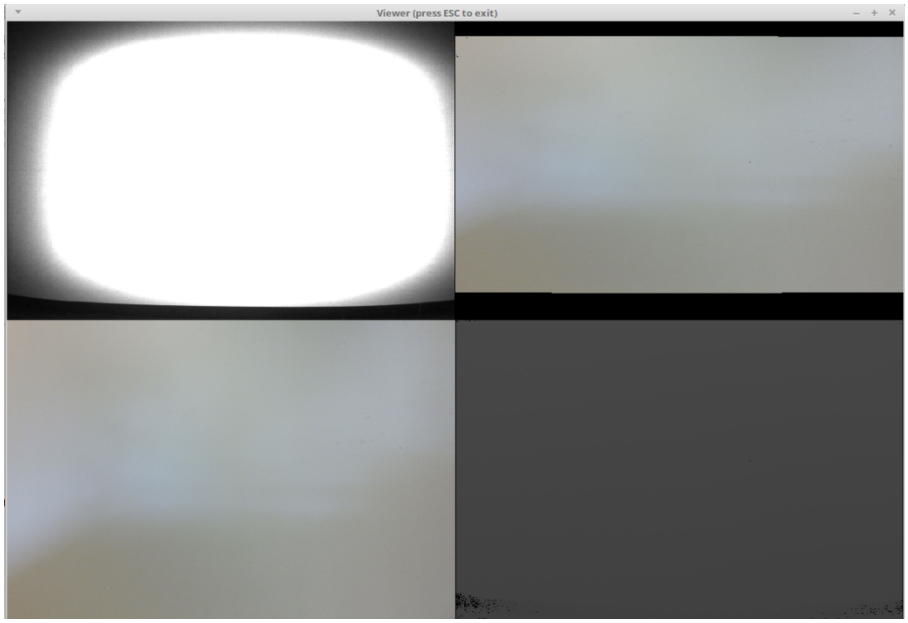


Figure 41: Application fenêtrée de la librairie Libfreenect2- Surface plane et parallèle à la Kinect v2

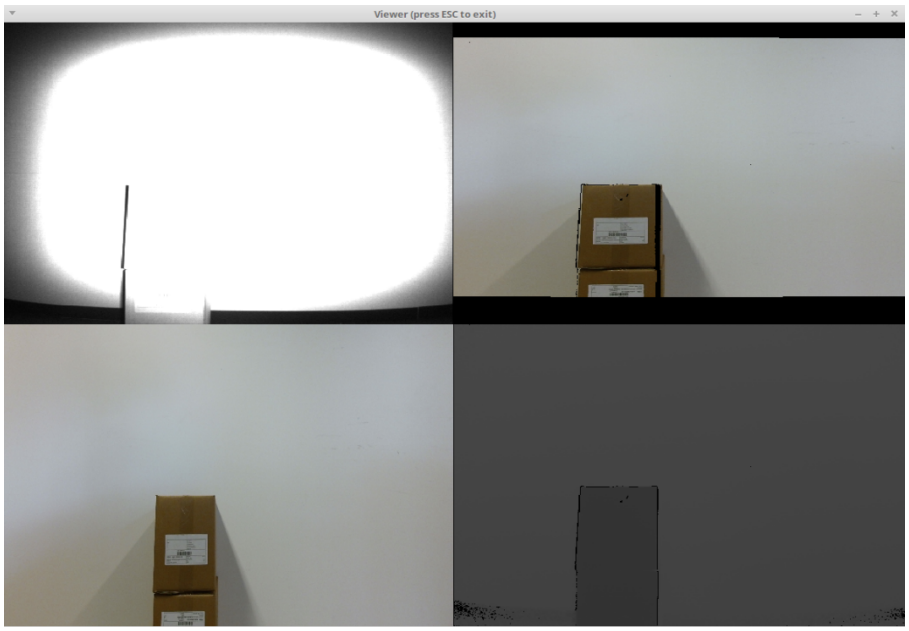


Figure 42: Application fenêtrée de la librairie Libfreenect2- Surface plane parallèle à la Kinect v2 avec une caisse placée dans le champ.

Le programme a été modifié pour pouvoir extraire les données des frames sous forme de fichier et les traiter avec dans le logiciel Matlab. La Figure 43 présente les résultats correspondant aux deux surfaces présentées à la Figure 41 et à la Figure 42. On peut remarquer la variabilité des mesures entre chaque pixel de l'ordre du millimètre, elle augmente significativement aux coins de la surface. Cela entre en concordance avec l'effet de lentille visible sur la vidéo de l'image infrarouge. On remarquera aussi que la quantité de pixels sans données sur ces figures (en mauve) est plus importante au droit des coins inférieurs de la surface et au droit des arrêtes de la caisse.

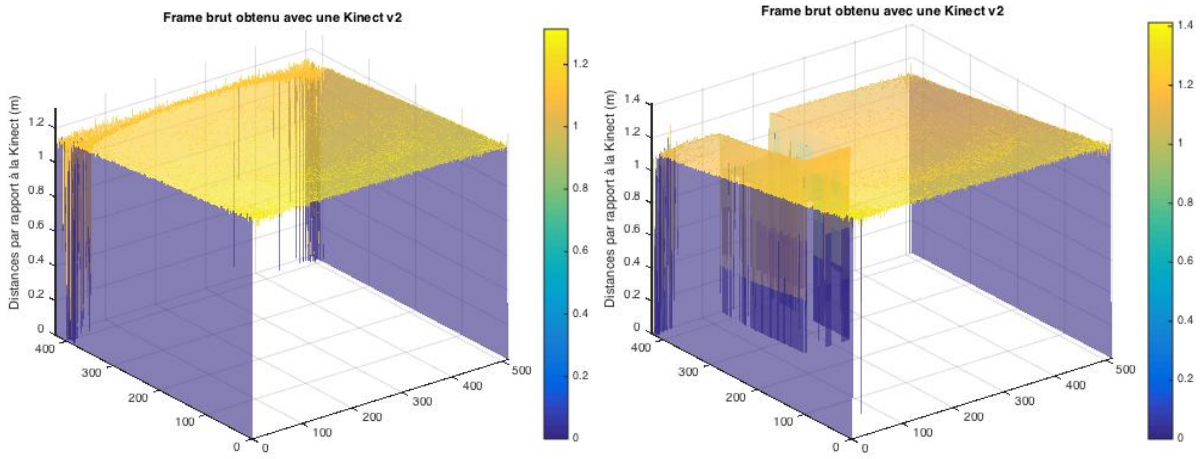


Figure 43: Frame brut représenté dans Matlab. A gauche, une surface captée plane, à droite surface plane avec un caisse au milieu inférieur de la surface

Sur 100 frames de profondeurs sauvegardés on obtient en moyenne 0,51 % de pixels sans valeurs. La présence dans le champ visuel d'un objet aux arrêtes saillantes, augmente le pourcentage de pixels sans données (0,72 %) avec une présence importance de ceux-ci au droit des arrêtes comme on peut le voir sur la Figure 43 avec la présence d'une caisse de 25cm x30cm. Les frames de distance ont une dimension de 512 pixels par 424 pixels.

L'analyse du code et les modifications effectuées sur de ce programme ont permis d'établir les étapes indispensables pour initialiser un streaming des frames par la Kinect v2 à l'aide de cette libraire :

1. Initialisation de deux classes, une pour trouver et ouvrir les appareils connectés et l'autre qui sert d'identificateur de la Kinect v2 connectée.
2. Enumération des dispositifs connectés
3. Ouverture de la Kinect sur base de son numéro de série
4. Configuration de la Kinect pour lui permettre d'envoyer des images
5. Début de la captation d'une surface
6. Réception des frames
7. Arrêter la captation d'une surface

Il faut noter que cette librairie et le programme exemple sont très bien structurés et documentés : l'initialisation, la captation et la réception de frame bruts se fait en quelques appels des fonctions.

2. La modification du programme SARndbox

A la suite de l'interaction possible avec la Kinect v2 par la librairie libfreenect2, la modification du programme SARnbox pour que la Kinect v2 soit supportée a été envisagée.

Tout d'abord, une liste de tous les appels au logiciel Kinect 3D Video Processing dans le programme SARndbox a été effectuée. Cette liste compte plus de 80 appels à des fonctions et classes. Les appels ne sont pas regroupés dans un seul fichier du programme. Le Tableau 8 présente les fichiers composants le programme principal avec l'indication s'il y a présence d'appel ou non au programme interagissant avec la Kinect.

Nom	Appel au logiciel Kinect
BathymetrySaverTool	X
CalibrateProjector	X
Config	
DEM	
DEMTool	
DepthImageRenderer	X
ElevationColorMap	

FindBlobs	
FrameFilter	X
GlobalWaterTool	
HandExtractor	X
LocalWaterTool	X
RainMaker	X
Sandbox	X
ShaderHelper	
SurfaceRenderer	X
Types	
WaterRenderer	
WaterTable2	

Tableau 8 : Fichiers comprenant des appels au logiciel Kinect 3D Video Processing

Tous ces appels se trouvent dans un nombre réduit de classes définies dans le logiciel Kinect. La liste de ces fichiers ainsi que leur fonction principale est présentée au Tableau 9.

Fichier	Fonction
DirectFrameSource.cpp	Classe intermédiaire pour les sources de frames directement connectées à la caméra
OpenDirectFrameSource.cpp	Classe pour l'ouverture d'un appareil sur base de son indice ou de son numéro de série sans connaitre le type de caméra
FrameBuffer.h	Classe de référence de frames couleur et de distance décodés
FrameSource.cpp	Classe de base d'objets qui créer un téléchargement de frames couleur et de distance
Camera.cpp	Classe pour représenter les aspects de d'interaction avec la Kinect.

Tableau 9 : Liste des fichiers du logiciel Kinect 3D Video Processing ulisée dans le logiciel SARndbox

Suite à l'analyse du code, les fonctions d'utilisation de la Kinect sont contenues dans le fichier `camera.cpp` avec 26 fonctions d'appels (Tableau 11), elles concernent l'ouverture, l'initialisation de la caméra et téléchargement des frames.

Le nombre important d'appels dans tout le programme, informe de la complexité des programmes SARndbox et Kinect 3D Video Processing. Le Tableau 10 présente la complexité de logiciels et librairies utilisées par des métriques obtenues avec l'outil SourceMonitor. L'Annexe 9 présente les détails pour les logiciels analysés dans cette étude à savoir SARndbox 2.3 et Kinect-3.2.

	SARnbox	Kinect 3.2	VRUI	libfreenect2
Fichiers	35	123	1397	72
Lignes	8 806	26 098	243 985	27 934
Ligne de code	4969	14015	123564	11379
% de commentaires	20,5	19,4	18,6	11,1
Classes	44	142	1775	233

Tableau 10: Complexité des logiciels utilisés

2.1 Utilisation du logiciel Kinect 3D Video Processing

Une fois la Kinect connectée à un port USB de la carte mère, la détection de la Kinect a été possible en compilant le programme `KinectUtil` pour la Kinect v2 avec le fichier spécifique `camerav2.cpp`.

La vérification de son fonctionnement a été effectuée en exécutant l'exécutable RawKinectViewer qui permet d'ouvrir une application fenêtrée avec les vidéos de la Kinect. De nouveau, l'exécution n'a pu aboutir à cause du retour de codes d'erreurs. Les investigations ont été menées et la cause est l'absence d'un fichier contenant une matrice de couleur dans le dossier du programme. Le bypass de cette opération a été envisagée sans succès. Le contact avec le développeur pour fournir la matrice manquante s'est relevé infructueux.

Etant donné le temps passé dans ces investigations infructueuses, les multiples appels dans le programme, et le stockage de données dans le programme sans en connaitre les tenants et aboutissants, cette piste d'utilisation du logiciel Kinect 3D Video Processing a donc été mise de côté.

2.2 Utilisation de la librairie LibFreenect2

L'étude de modification du programme SARndbox, s'est poursuivie en essayant d'utiliser les fonctions de librairie dynamique Libfreenect2 à la place de ceux de la librairie Kinect 3D Video Processing.

Cela a débuté par essayer de trouver une correspondance des appels présents dans le fichier `camera.cpp` avec les fonctions disponibles dans la librairie Libfreencet2. Les résultats de cette étude sont présentés au Tableau 11.

Nom de la fonction	Libfreenect2	Utilité
`void* colorDecodingThreadMethod(void);`		Thread[5] pour décoder un frame brut couleur
`void* depthDecodingThreadMethod(void);`		Thread pour décoder un frame brut de distance
`void*`		Thread pour décoder un frame compressé

[5] Processus informatique qui est exécuté en parallèle de l'exécution d'autre processus

compressedDepthDecodingThreadMethod (void);		
void initialize(USB::DeviceList* deviceList =0);	X	Initialisation de la caméra Kinect
Camera(libusb_device* sDevice);		Création l'objet caméra à partir de l'identificateur du port USB
Camera(size_t index =0)	X	Ouverture de la Kinect à partir de son indice
Camera(const char* serialNumber);		Ouverture de la Kinect à partir de son numéro de série
virtual DepthCorrection* getDepthCorrectionParameters(void);		Obtention des paramètres de correction de profondeur
virtual IntrinsicParameters getIntrinsicParameters(void);	X	Obtention des paramètres internes de la Kinect
virtual const unsigned int* getActualFrameSize(int sensor) const;	X	Obtention de la taille en pixels des frames
virtual DepthRange getDepthRange(void) const;		Obtention de l'intervalle de distances
virtual void startStreaming(StreamingCallback* newColorStreamingCallback,StreamingCallback* newDepthStreamingCallback);	X	Initialisation du chargement des frames
virtual void stopStreaming(void);	X	Arrêt du téléchargement des frames
virtual std::string getSerialNumber(void);	X	Obtention du numéro de série de la Kinect
void getCalibrationParameters(CalibrationParameters& calib);		Obtention des paramètres de calibration de la Kinect
void setFrameSize(int camera,FrameSize newFrameSize);		Imposition de la taille de frames en pixels
FrameSize getFrameSize(int camera)	X	Obtention de la taille de frames en pixels
void setFrameRate(int camera,FrameRate newFrameRate);		Imposition de la fréquence de captation des frames
FrameRate getFrameRate(int camera) cons	X	Obtention de la fréquence de captation des frames
unsigned short getIrIntensity(void)		Obtention de l'intensité infrarouge
void setIrIntensity(unsigned short newIrIntensity);		Imposition de l'intensité infrarouge
bool supportsNearMode(void) const		Caméra en mode proche ou non
bool getNearMode(void) const		Imposée le mode proche de la caméra
unsigned int getSharpening(void);		Obtention des paramètres de netteté de couleur
unsigned int getSharpening(void);		Imposition des paramètres de netteté de couleur

Tableau 11: Liste des fonctions du fichier camera.cpp avec leur concordance dans la librairie Libfreenect2

On peut remarquer que seuls les appels principaux ont une correspondance. Tous les appels qui permettent d'imposer à la Kinect une valeur d'un paramètre dans son fonctionnement comme par exemple le taux de captation ou la taille des frames couleur ou un mode de fonctionnement, … n'ont par contre aucune correspondance. Pour y pallier, trois alternatives ont été envisagées : essai de contourner l'utilisation de certaines fonctions, remplacement par du code nouveau ou l'utilisation d'une librairie plus complète comme OPenNi2. Les deux premières alternatives furent retenues.

La première étape consistait à insérer la librairie Libfreenect2 dans les sources du programme, ensuite remplacer le premier appel et compiler le programme. Après avoir passé un temps significatif sur cette étape nous avons dû renoncer à cette étape. En effet, la manière dont la compilation a été faite dans le programme SARnbox fait appel à des notions avancées de makefile dans l'environnement LINUX. De plus, aucun débugger ne fonctionnait pour pourvoir exécuter le programme par partie et analyser les erreurs. Nos connaissances en informatique n'étant pas suffisantes, poursuivre dans cette voie aurait été consommatrice de temps pour utiliser une Kinect v2 sans garantie de résultat et risque de ne pas atteindre les objectifs de ce travail.

Par conséquence, la modification du logiciel SARndbox pour intégrer le version 2 a été interrompue afin de pouvoir dans le temps imparti mettre en place un bac à sable fonctionnel et analyser en détails l'écoulement calculé en utilisant la version 1.

Chapitre 5 Mise en place d'un bac à sable à réalité augmentée

L'adaptation du système existant à la Kinect seconde version n'ayant été possible, un bac à sable fonctionnel a été mis en place avec une Kinect première génération. Ce chapitre détaille les étapes d'installation ainsi que l'utilisation du programme et du bac à sable.

1. Installation et fonctionnement

1.1 Procédure d'installation

La première étape est la mise en place du bac et de la potence de manière à ce que le champ visuel de la Kinect et celui du projecteur couvrent l'intérieur du bac. Lors de cette étape, les dimensions finales de la potence ont été déterminées. A cause du au décalage de l'image du projecteur par rapport au centre de la lentille, la potence a dû être écartée du bac comme cela est visible sur les plans à l'Annexe 4.

La deuxième étape consiste en l'installation et la configuration du logiciel SARndbox.

Après l'installation sur l'ordinateur de la version Linux Mate 64 bits, les principales étapes de cette procédure sont :

- Installation des logiciels : Vrui VR Toolkit en version 4.2-006, Kinect 3D Processing en version 3.2 et SARndbox en version 2.3.
- Connexion de la Kinect v1
- Obtention de l'équation du plan capté par la Kinect ainsi que de la position des coins de cette surface
- Connexion du projecteur et alignement.
- Calibration du projecteur et de la Kinect.
- Exécution du programme

Cette installation ne s'est pas déroulée sans problèmes, notamment à cause de lacunes dans les explications d'installation ou détails manquants liés à la configuration. La procédure traduite en français est disponible et corrigée qui a été appliquée est fournie à l'Annexe 6.

La Figure 44 présente le bac à sable en fonctionnement un fois le programme lancé.

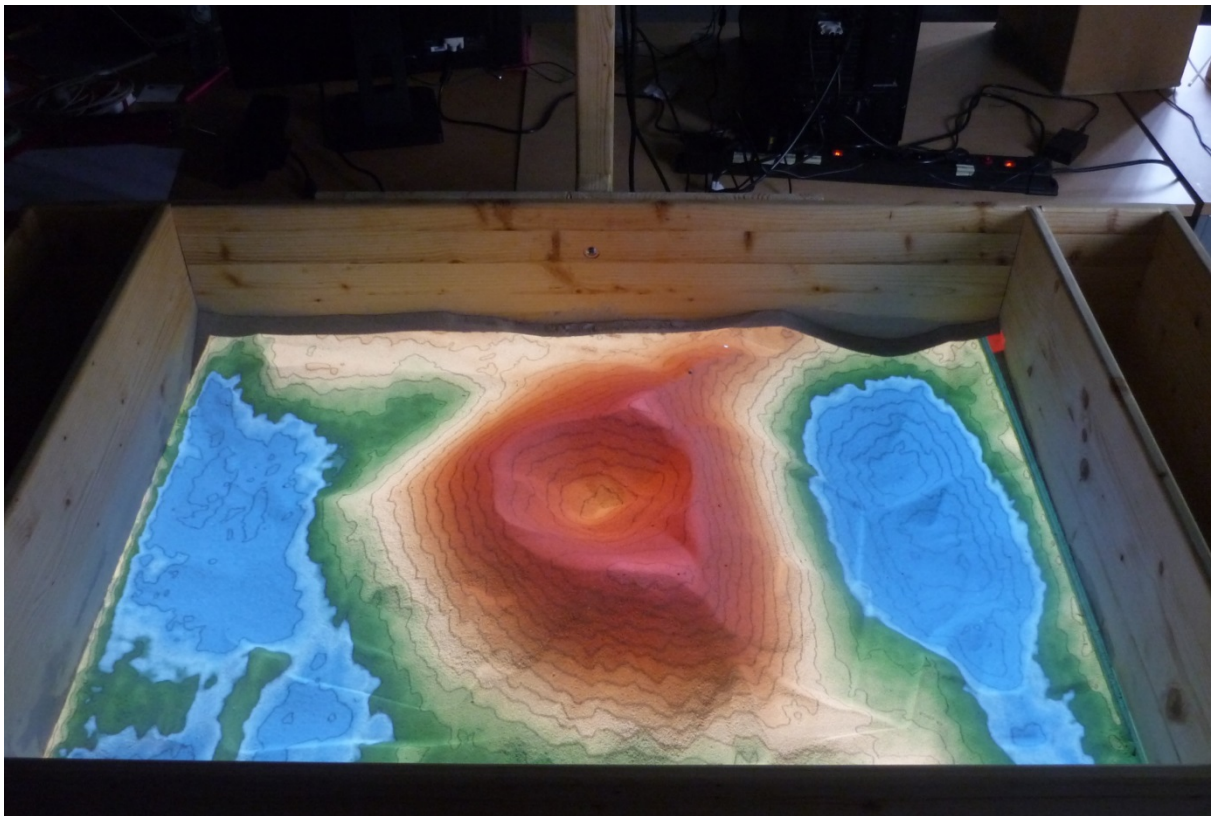


Figure 44: Bac à sable à réalité augmentée en fonctionnement

1.2 Options d'exécution

Le lancement de l'application se fait dans une fenêtre terminal en se mettant dans le bon répertoire puis en lançant le programme par les commandes suivantes.

```
cd ~/src/SARndbox-2.3
./bin/SARndbox -uhm -fpv
```

Dans cette dernière ligne de commande les instructions –uhm et –fpv sont des options d'exécution. La première spécifie que le rendu coloré de la surface topographique est activé et la seconde, utilise la calibration du projecteur et de la Kinect réalisée en phase d'installation afin que le rendu projeté sur le sable se superpose parfaitement à l'image captée.

La liste des autres options d'exécution est fournie au Tableau 12.

Options	Fonctions
-h	Affichage du message d'aide
-c <camera index>	Sélectionne la caméra de l'indice donné. **Par défaut :** indice 0
f <frame file name prefix>	Lecture d'une vidéo préenregistrée à partir d'une paire de buffer couleur/distances

	contenue dans le fichier spécifié.
-s <scale factor>	Facteur d'échelle entre le bac à sable et le terrain simulé. **Par défaut** : 100 (1/100, 1cm dans la Sandbox =1m sur le terrain)
-slf <sandbox layout file name>"	Chargement du fichier contenant les paramètres de configuration. **Par défaut :** CONFIG_CONFIGDIR
-er <min elevation> <max elevation>	Imposition de l'intervalle d'élévation de la surface de sable relatif à la surface plane en cm. **Par défaut :** Intervalle d'élévation de la carte de couleur
-hmp <x> <y> <z> <offset>	Imposition de l'équation du plan utilisé pour la coloration des hauteurs
-nas <num averaging slots>	Imposition du nombre de frame pour établir le frame de moyennage utilisé dans l'opération de filtrage des frames. **Par défaut :** 30
-sp <min num samples> <max variance>	Imposition des paramètres de filtrage des frames avec le minimum d'échantillon valide et la variance maximale avant convergence. **Par défaut :** 10 2
-he <hysteresis envelope>	Imposition de la taille de l'enveloppe d'hystérésis utilisé dans l'opération de filtrage des frames. **Par défaut** : 0,1
-wts <water grid width> <water grid height>	Imposition de la largeur et de la hauteur (en pixels) de la grille utilisée pour la simulation d'écoulement. **Par défaut** : 640 480 pour la Kinect 1.
-ws <water speed> <water max steps>	Imposition de la vitesse relative de la simulation d'écoulement et du nombre de pas de simulation par frame. **Par défaut** : 1,0 30
-rer <min rain elevation> <max rain elevation>"	Imposition de l'intervalle d'élévation du niveau du nuage de pluie relatif à la surface plane en cm. **Par défaut :** Au-dessus de l'intervalle de la carte de couleur
rs <rain strength>	Imposition de l'intensité de pluie locale ou globale en cm/s. **Par défaut** : 0,25
-evr <evaporation rate>	Imposition du taux d'évaporation en cm/s. **Par défaut** : 0,0
-dds <DEM distance scale>	Facteur d'échelle en cm pour fonctionner avec le modèle d'élévation digital.
-wi <window index>	Imposition de l'indice de l'écran pour le rendu auquel les paramètres de configuration sont

	appliqués. **Par défaut** : 0
-fpv [projector transform file name]	Fixe la transformation pour que la Kinect et le projecteur soient alignés comme défini dans le fichier de transformation du projecteur. **Fichier de transformation par défaut :** CONFIG_CONFIGDIR
-nhs	Désactiver les ombres de montagnes.
-uhs	Activer les ombres de montagnes.
-ns	Désactiver les ombres.
-us	Activer les ombres.
-nhm	Désactiver la colorisation de la topographie.
-uhm [elevation color map file name]	Activer la colorisation de la topographie avec la carte des couleurs dans le fichier spécifié. **Fichier de colorisation par défaut :** CONFIG_CONFIGDIR
- ncl	Désactiver la représentation des lignes topographiques.
-ucl [contour line spacing]	Désactiver la représentation des lignes topographiques et imposition de la distance (en cm) d'élévation entre les lignes. **Par défaut :** espacement de 0,75
-rws	Représentation de la surface d'eau comme une surface géométrique.
-rwt	Représentation de la surface d'eau comme une texture.
-wo <water opacity>	Imposition de la profondeur à laquelle l'eau apparaît comme opaque **Par défaut :** 2,0 cm

Tableau 12 : Liste de options d'exécution du logiciel SARndbox

2. Fonctionnalités principales

Une fois lancé, le logiciel SARnbox met à jour en permanence la topographie captée. A partir de cette configuration de base, plusieurs opérations peuvent être effectuées, elles sont présentes dans un menu principal accessible en maintenant enfoncé une touche quelconque du clavier et en choisissant ensuite le pointeur de la souris sur l'option retenue. Il faut noter que dans la suite chaque fois que cette touche est activée, la fonction du programme est activée. Nous présenterons dans la suite trois fonctionnalités :

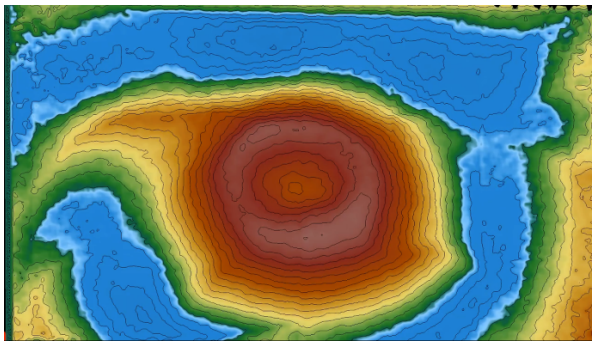
- Création de pluie locale ou globale
- Utilisation d'un modèle d'élévation digitale
- Transformation du rendu de l'eau en lave.

2.1 Création d'une pluie

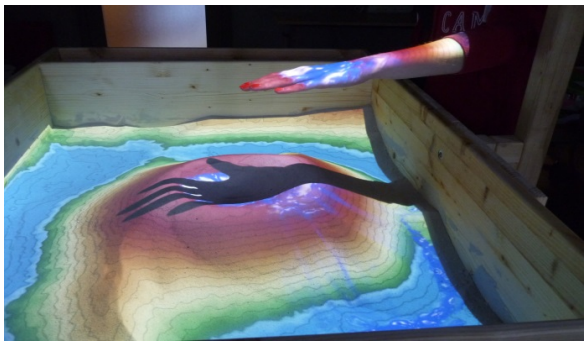
Deux sortes de pluie peuvent être générées par la Sandbox. L'une est locale et basée sur la détection d'une main ouverte à au moins 15cm de la topographie. Une pluie est alors appliquée sur la topographie sur une surface circulaire dépendant du diamètre de l'objet détecté. La seconde pluie est globale et appliquée sur toute la surface de sable. L'intensité de la pluie dans les 2 cas est 0,25cm/s qui correspond à l'intensité par défaut. Elle peut être modifiée par l'utilisateur lors de l'exécution du programme.

Tant que la touche d'accès à la fonction pluie globale est enfoncée l'eau s'écoulera sur le modèle. Pour enlever de l'eau du modèle, il suffira de maintenir enfoncée la touche associée à la configuration de l'outil pour assécher le modèle. Le menu secondaire accessible grâce au bouton de droite de la souris permet d'accéder aux paramètres d'écoulement et de les modifier. Ils permettent de modifier le terme d'atténuation, la fréquence de captation d'un frame, la vitesse relative de représentation de l'écoulement et le nombre maximal de pas de temps par mise à jour de bathymétrie.

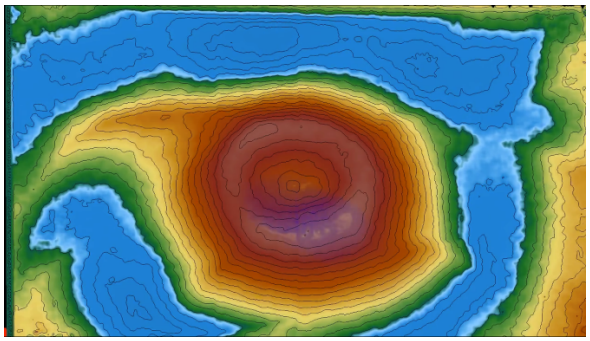
La Figure 45 présente l'évolution d'une pluie locale générée pendant toute la durée de simulation à partir de la présence de d'une main au-dessus du bac.



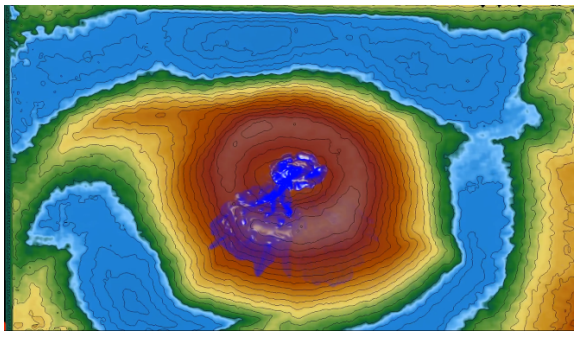
a) Topographie initiale



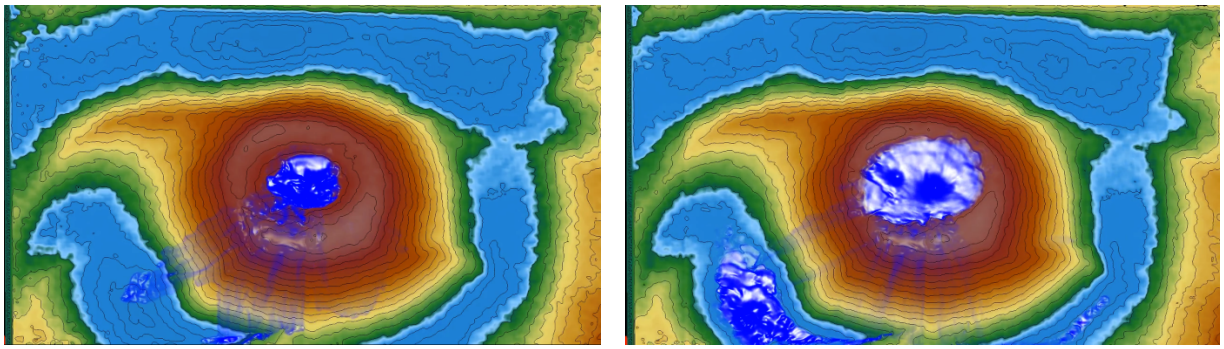
b) Placement de la main au-dessus du bac



c) Initialisation de la pluie : Temps=0 s



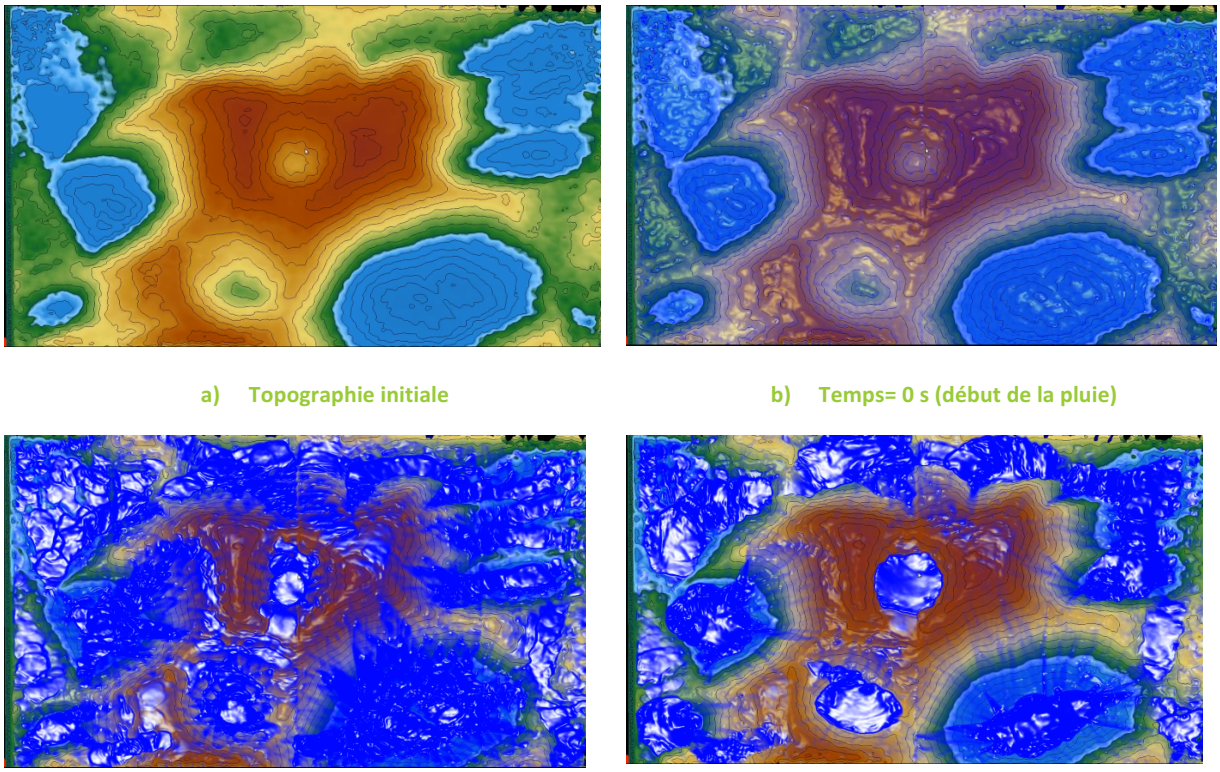
d) Temps= 10s



e) Temps=15s f) Temps = 45s

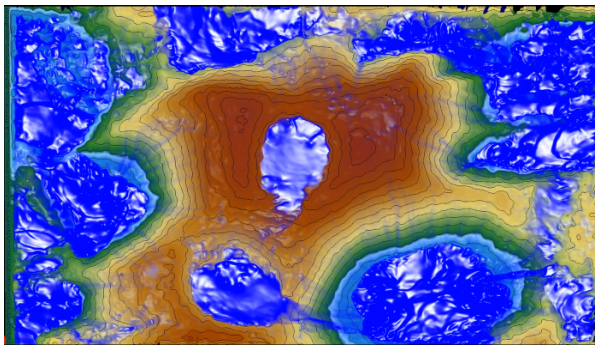
Figure 45: Évolution du rendu d'une pluie locale d'une durée de 3 secondes d'intensité 0,25cm/s

La Figure 46 présente l'évolution d'une pluie globale générée pendant trois secondes.

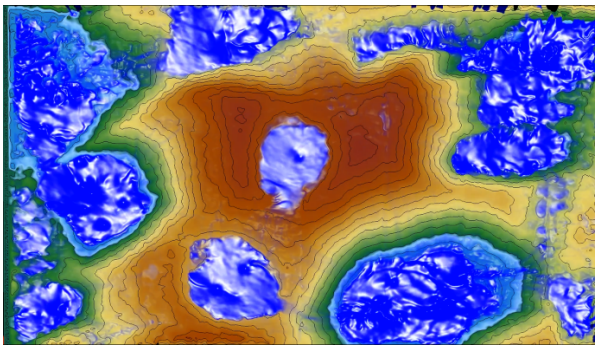


a) Topographie initiale b) Temps= 0 s (début de la pluie)

c) Temps= 3 s (fin de la pluie) d) Temps= 6s



e) Temps=10s



f) Temps=15s

Figure 46: Évolution du rendu d'une pluie globale d'une durée de 3 secondes d'intensité 0,25cm/s

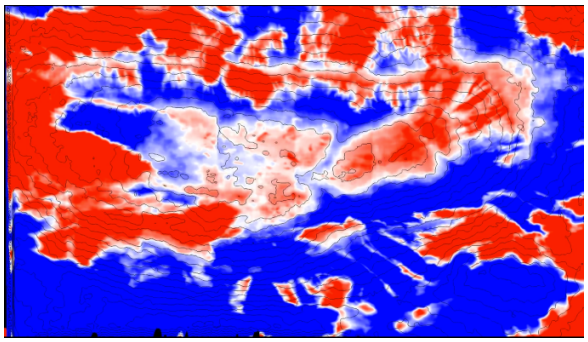
2.2 Utilisation d'un modèle d'élévation digital

Une autre fonctionnalité de la Sandbox est de projeter un modèle d'élévation digital d'un terrain réel sur le sable. La projection sur le sable est alors coloriée en trois couleurs : rouge où la hauteur de sable est trop faible, bleue où la hauteur de sable est trop importante et blanc où la quantité est suffisante. Ceci permet à l'utilisateur de modeler dans le sable la topographie à reproduire afin de pouvoir par la suite lui appliquer un écoulement puisque qu'il est possible de passer d'une représentation pour le modelage à la présentation normale.

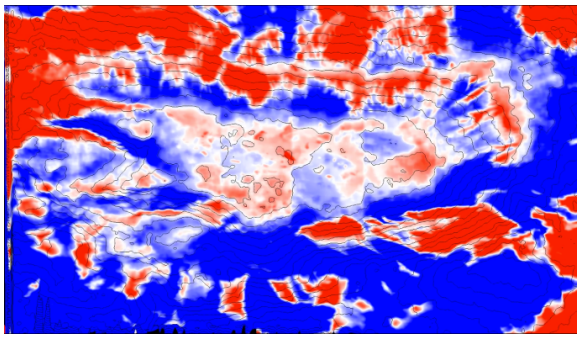
Le chargement d'un DEM, s'effectue par le menu principal de la Sandbox en choisissant l'option « Show DEM » le choix du fichier DEM. L'extension du fichier doit être .grid qui correspond à un format binaire pour une image raster. Ce format est binaire et de type little-endian avec les 2 premières entrées contenant le nombre de lignes et colonnes du DEM exprimées en format unsigned integer 4 byte. Les 4 entrées suivantes sont l'étendue spatiale exprimée comme les coordonnées spatiales des 4 sommets en format IEEE 754 nombre en float. L'entrée suivante est constituée des élévations de chaque point dans un format de type float.

Un ajustement vertical du DEM ainsi que l'échelle verticale et spatiale peuvent alors être modifiés, en accédant à la fonction « Save Input Graph » du sous menu « Devices » du menu Vrui dans le menu secondaire rendant à ce jour l'utilisation de l'outil DEM complexe.

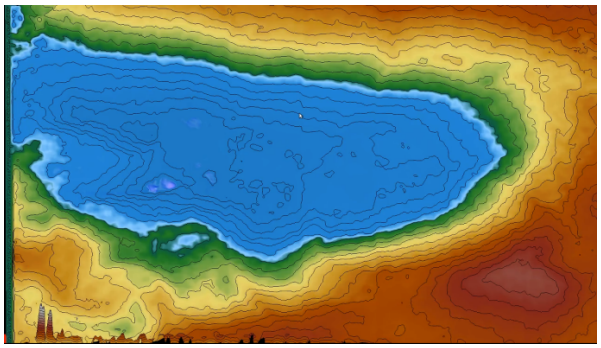
Un modèle d'élévation Digital du lac Tahoe en Californie a été réalisé avec le bac à sable, la Figure 47 présente les images des différentes étapes ainsi qu'une carte topographique de ce lac à l'image d).



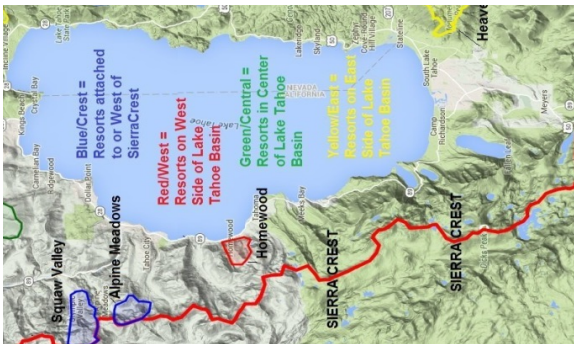
a) Chargement du DEM



b) Modification du relief dans le sable



c) Représentation normale de la surface b



d) Carte topographique du lac [18]

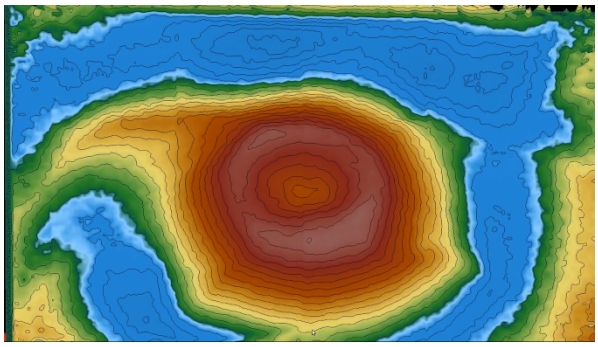
Figure 47: Utilisation d'un modèle d'élévation digitale du lac Tahoe en Californie

Il est aussi possible de sauver la topographie créée dans le sable dans un format .dem via l'outil « Save Bathymetry » dans le menu principal.

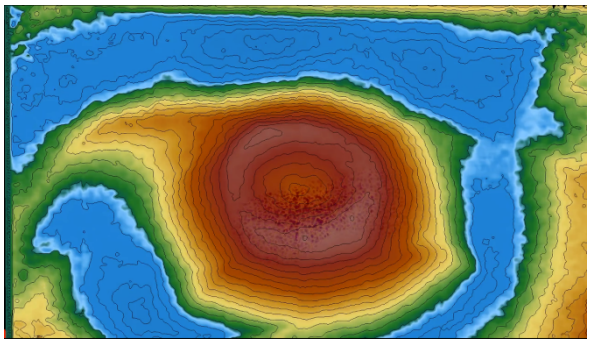
2.3 Changement du fluide en lave

Il est possible de changer l'aspect du fluide dans la Sandbox, en changeant la matrice de couleur dans le shader SurfaceAddWaterColor.fs. Cette procédure de modification est expliquée à l'Annexe 7. Il est important de noter que cela n'affectera que l'aspect visuel dans la Sandbox et donc pas le comportement. Si on veut rendre le rendu plus réaliste pour de la lave, la vitesse relative devra être diminuée ou il faudra modifier les équations d'écoulement pour être précis.

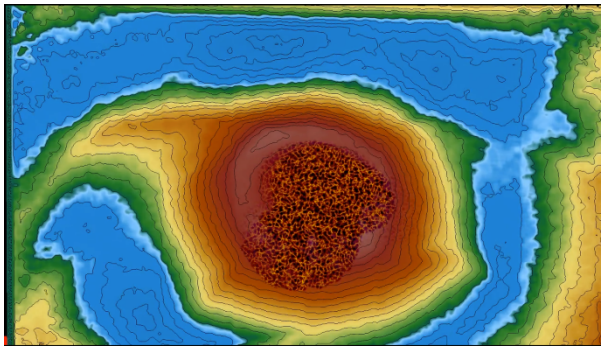
La Figure 48 présente le changement de texture avec le paramètre WaterSpeed imposée à 0,1. Ce paramètre est la vitesse relative de la simulation d'écoulement, il peut être vu comme la vitesse de représentation de l'écoulement, si cette valeur est inférieure à 1 alors l'écoulement est ralenti par rapport à l'écoulement réellement calculé. Dans le cas contraire, l'écoulement représenté est accéléré par rapport à l'écoulement calculé.



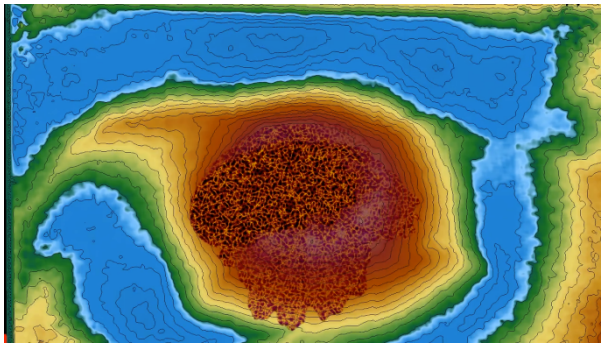
a) Topographie initiale



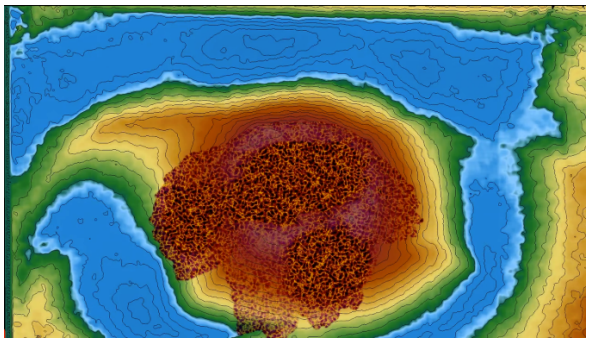
b) Début d'éruption : Temps=0s



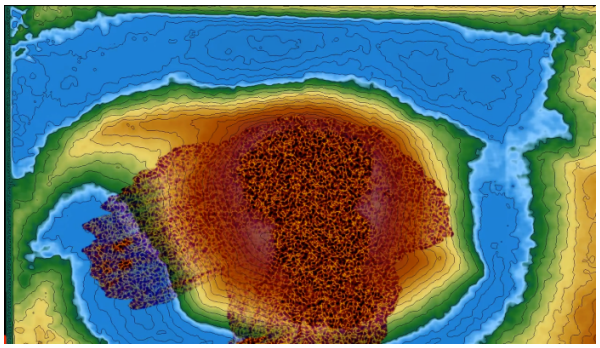
c) Temps=10s



d) Temps=20s



e) Temps=30s



f) Temps=40s

Figure 48: Représentation d'une éruption volcanique

Chapitre 6 Extraction et comparaison de l'écoulement

Une fois, le bac à sable à réalité augmentée fonctionnel obtenu, l'étape suivante consiste à analyser les résultats de l'écoulement calculé de manière qualitative et quantitative. Afin de réaliser cette analyse, ce chapitre détaille la modification du code informatique du logiciel SARndbox en vue de ne plus utiliser la Kinect v1 pour la captation mais d'imposer au programme une topographie définie. La seconde partie tentera de comparer les résultats du programme à celui d'un programme spécialisé dans le calcul d'écoulement, Wolf2D.

1. Modification du programme

1.1 Imposition d'une topographie connue

La modification pour insérer une topographie a été envisagée de 3 manières :

- La modification de la topographie lors l'étape de filtrage des frames
- La création d'un shader pour la modification de la bathymétrie
- L'imposition d'une matrice lors de la fonction de mise à jour de bathymétrie

La première n'a pu aboutir car dans cette étape, la lecture de la topographie cause aussi des transformations d'échelle des données et des transformations de rendu sont effectuées pour garantir un rendu visuel satisfaisant mais correspondant à d'autres données obtenues de la caméra. La seconde possibilité a été faiblement investiguée car les appels et les structures de données n'étaient pas explicites. Comme aucun débugger ni environnement n'était disponible, la mise au point pour le codage d'un shader risquait de prendre du temps.

C'est donc la troisième méthode qui a été utilisée. Dans le code existant, il existe une autre méthode `updatebathymetry()` qui possède un argument supplémentaire correspondant à un pointeur vers une matrice. Dans cette fonction, la matrice sera passé en argument de la texture de bathymétrie `bathymetryTextureObjet.` En affectant les valeurs de cette matrice aux données de topographie, la bathymétrie imposée est utilisée dans le calcul d'écoulement à la place de la captation de la caméra.

Cependant, il est utile de noter que cette imposition n'implique pas que la Kinect peut être éteinte ou non opérationnelle. En effet, le lancement du programme ne fonctionne que si la Kinect est détectée au démarrage du programme et si elle reste allumée en permanence. De plus, le rendu visuel de la Sandbox de la topographie correspondra à celui capté par la Kinect même avec une imposition de topographie. En effet le rendu de topographie s'effectue sur base des données captées. Cependant, l'eau représentée correspond bien à l'écoulement calculé sur base de cette imposition. Pour éviter que le rendu visuel superpose une topographie non cohérente avec l'écoulement calculé, les paramètres d'exécution du programme devront être modifié en vue de

n'afficher que l'écoulement d'eau sans la représentation des courbes de niveaux (- ncl) et la représentation de la topographie (-nhm).

1.2 Sauvegarde des résultats

Suite à l'imposition d'une topographie définie, le rendu visuel dans la Sandbox semblait cohérent et en accord avec la topographie imposée. Il a donc été envisagé de sauvegarder les résultats générés lors du calcul de l'écoulement pour la comparaison. Cette opération consiste à la lecture des données des pixels d'une texture s'ils ont été stockés dans un frame buffer sous forme de matrices. Deux étapes sont indispensables pour effectuer cette opération du GPU vers le CPU :

1. Spécification du buffer de couleurs à lire.
2. Lecture des données. Afin de permettre le stockage de données, il est nécessaire d'indiquer les coordonnées du premier pixel à lire dans le buffer correspondant au pixel en bas à gauche du rectangle de pixels à lire, les dimensions du rectangle à lire, le format des pixels, le type de données du pixel dans le buffer et la variable préalablement définie pour le stockage de pixels.

La lecture d'un frame buffer s'effectue par ligne en commençant par la lecture de la valeur de 3 composantes du pixel en bas à gauche du rectangle défini et en remontant le rectangle en ligne en commençant toujours par le pixel de gauche. Pour analyser une seule composante de couleurs des pixels d'un rectangle, un traitement du vecteur obtenu lors de lecture devra être réalisé pour extraire une composante sur trois.

1.3 Modification des textures de bathymétrie

Lors des premières lectures des résultats du calcul de l'écoulement, il est apparu que les résultats obtenus dans les matrices ne correspondaient pas à ce qui devait être obtenu malgré un rendu visuel adéquat avec notamment deux lignes de valeurs totalement incohérentes.

Une analyse plus en détail des différentes étapes de calcul a montré que le problème venait de l'étape de moyennage de la bathymétrie centrée sur les sommets pour obtenir la bathymétrie au centre de la cellule.

Tout d'abord, initialement la taille de la texture de bathymétrie est inférieure à la taille de la texture quantité 639 x479 pixels pour la première contre 640 x480 pixels pour la seconde. La texture de bathymétrie est donc centrée sur la texture de quantité. De plus, la valeur des pixels de la texture de bathymétrie correspond à la bathymétrie au sommet d'une cellule contrairement à la texture quantité où les valeurs sont exprimées au centre de la cellule. La superposition des 2 textures est disponible à la Figure 49a avec des textures de plus faibles dimensions.

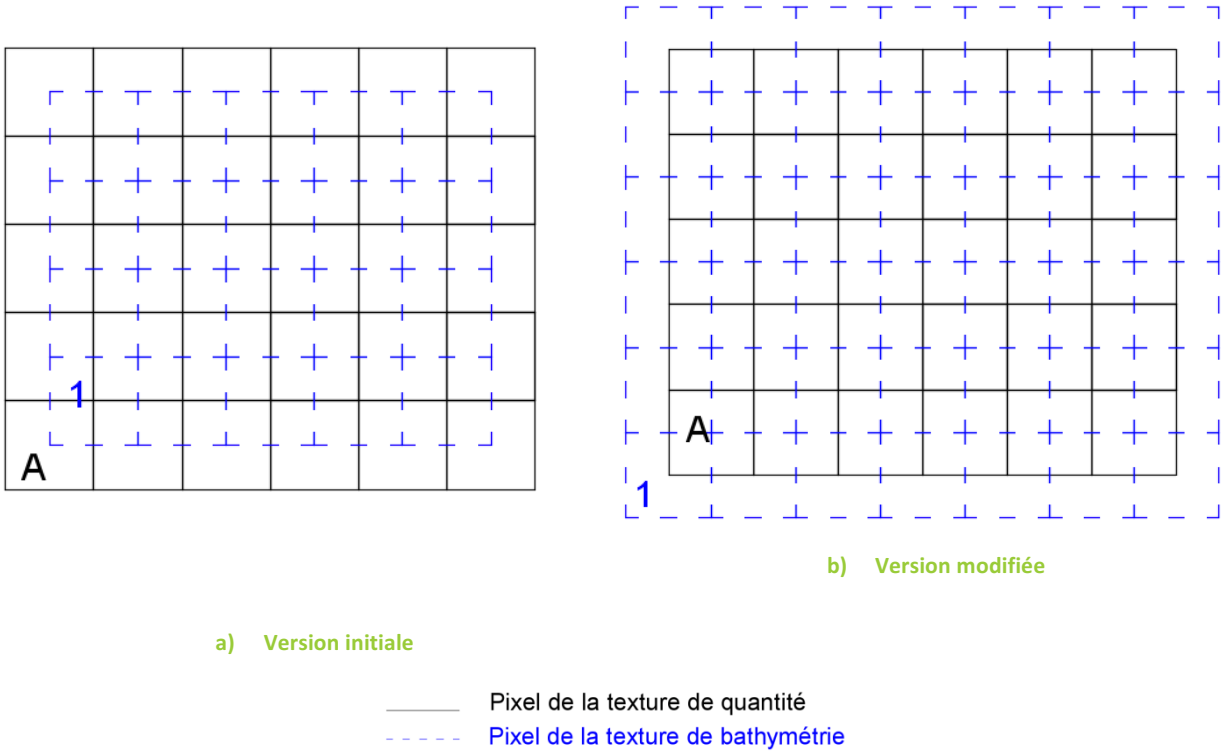
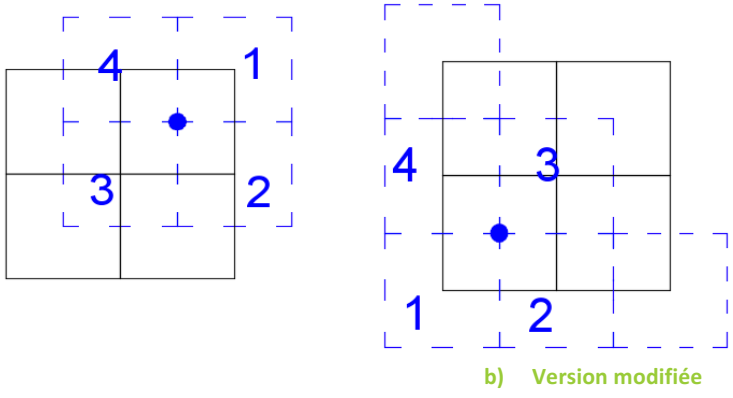


Figure 49: Superposition de la texture de bathymétrie et de la texture de quantité a) dans la version initiale SARndbox et b) version modifiée du code informatique du programme.

La texture de bathymétrie est utilisée lors de l'opération de moyennage pour obtenir une bathymétrie centrée sur un cellule et pour obtenir la hauteur d'eau au centre d'une cellule puisque l'inconnue de hauteur d'eau (h) a été remplacée par l'élévation de la surface libre $(w = h + B)$ dans le schéma de discrétisation temporelle. De plus, cette opération est la moyenne du pixel sur lequel on calcule et des trois pixels à ses cotées vers le bas comme exprimé à la Figure 50a avec le pixel 1 comme pixel de base et les pixels 2, 3 et 4 comme pixels utilisés pour la moyenne.



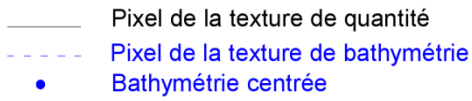


Figure 50 : Représentation de pixels utilisés pour le calcul de la moyenne de la bathymétrie

L'utilisation d'un shader est effectuée sur chaque pixel du rectangle défini en commençant par le pixel en bas à gauche. Dans le cas, par exemple du pixel dans le coin gauche, cette opération de moyennage n'est pas exacte mais aussi dans le cas d'un pixel au bord ou au coin du domaine. En effet, si on analyse le premier pixel parcouru soit le pixel A sur la Figure 49 de la texture quantité et le pixel 1 dans la texture bathymétrie, on remarque que pour le calcul de la moyenne les pixels de bathymétrie on va chercher la valeur de pixels non existants en dehors du domaine défini. La valeur prise pour ces pixels est la valeur des pixels au bord de domaine, cette norme est dû au type de texture utilisé soit des textures ARB. Dans le cas du pixel A, la bathymétrie centrée correspond donc à la valeur du pixel 1. Cette modélisation amène donc des erreurs qui se propagent dans le code.

Une correction des dimensions de la texture de bathymétrie et du moyennage a donc été apporté en modifiant le code informatique.

Dans un premier temps, les dimensions de la texture de bathymétrie ont été changées avec une taille de 641 par 481 pixels, cela permet que la texture quantité soit centrée sur la texture de bathymétrie comme représenté à la Figure 49 b.

Dans un second temps le calcul de la moyenne de bathymétrie a été changé en effectuant le moyenne du pixel sur lequel on calcule et des trois pixels à ces cotés vers le haut comme exprimé à la Figure 50b avec le pixel 1, le pixel de base et les pixels 2, 3 et 4, les pixels utilisés pour la moyenne. Ce changement empêche donc d'aller chercher des pixels non définis.

Les modifications ont donc été apportées aux shaders qui utilisent la bathymétrie. Les shaders modifiés pour le calcul d'écoulement sont :

- Water2BathymetryUpdateShader
- Water2BoundaryShader
- Water2SlopeFluxAndDerivativeShader
- Water2AdaptShader

2. Analyse des résultats du programme SARndbox

Grâce aux modifications présentées, les premiers résultats du calcul de l'écoulement paraissaient plus cohérents. L'analyse détaillée du calcul d'écoulement a pu être réalisée, elle a pour objectif de comparer trois critères :

- L'influence du terme d'atténuation
- L'influence des conditions aux limites imposées.
- L'influence du facteur d'échelle

Pour réaliser cette étude, un cas de base a été imposée à la bathymétrie. Le Tableau 13 présente ses principales caractéristiques.

Caractéristiques	Valeurs
Bathymétrie	Nulle partout
Vecteur quantité initiale	- Hauteur d'eau nulle sur les cellules sauf 0,1m sur les cellules comprises entre les pixels [284, 385] selon x et [189,286] selon y - Débits spécifiques selon x et y nuls
Conditions aux limites aux bords	- Hauteur d'eau nulle - Débits spécifiques selon x et y nuls
Critère de Courant	0,5
Taille de la texture de quantité	Selon x : 640 pixels Selon y : 480 pixels
Taille des cellules (m)	Selon x : 0,00150353 Selon y : 0,00146828
Temps de simulation	60 secondes
Temps de sauvegarde	Toutes les secondes
Échelle	1 /1

Tableau 13: Caractéristiques du cas de base

2.1 Facteur d'atténuation

Le paramètre d'atténuation est un paramètre qui apparait lors du calcul des pas d'intégration temporelle comme multiplicateur des débits spécifiques dans la direction de x et de y. Dans la version initiale du programme, sa valeur est fixée à 0,99. Pour étudier l'influence de ce paramètre, la comparaison des résultats a été effectuée avec une valeur de 0,6 dans le premier cas et une valeur de 1,0 dans le second. Ce facteur d'atténuation n'a aucune signification physique mais pourrait être assimilé à du frottement numérique.

La Figure 51 présente l'évolution temporelle de la hauteur d'eau et du volume d'eau au centre du pixel à la position dans la texture (200,100). Ce pixel est donc situé en dehors de la surface initiale où on a imposé de l'eau.

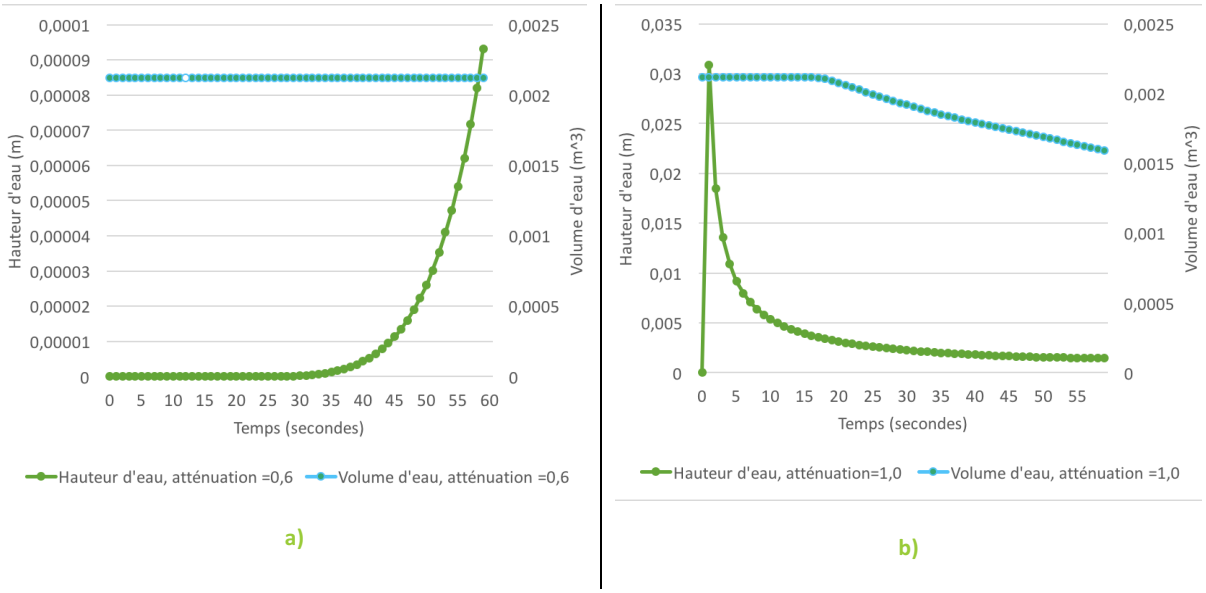
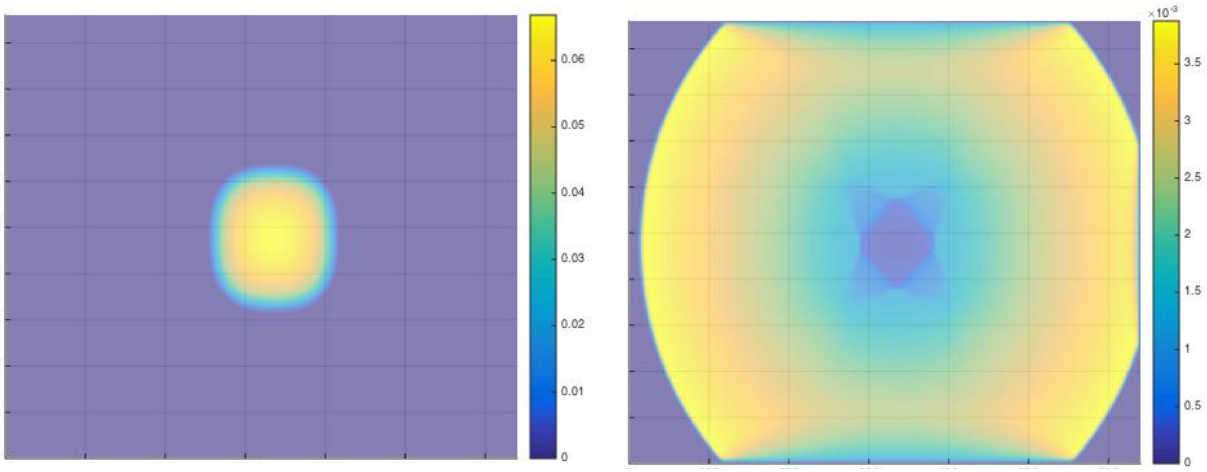


Figure 51 : Evolution de la hauteur d'eau et du volume d'eau d'un pixel en fonction du temps pour un facteur d'atténuation = 0,6 (a) 1,0 (b)

On remarque qu'après 45 secondes le front d'eau arrive au pixel proche de la surface d'eau imposée initialement dans le cas d'un facteur d'atténuation de 0,6 alors que dans le cas d'une valeur de 1,0, l'eau arrive instantanément au pixel et la hauteur d'eau diminue par la suite car aucun frottement n'est supposé, l'écoulement se poursuit. Il est à noter que la hauteur d'eau est très faible avec un coefficient de 0,6 par rapport à la hauteur d'eau dans le second cas. Cependant, la hauteur d'eau augmente dans le cas de d'un facteur égal à 0,6. Ceci peut être expliqué de la manière suivante : comme l'écoulement arrive progressivement à ce point la hauteur d'eau augmente, on s'attend à ce que cette hauteur d'eau diminue dans la suite de la simulation comme elle continuera de s'écouler sur le domaine.

Au niveau du volume d'eau, il reste constant dans le cas d'un coefficient d'atténuation égale à 0,6 dans le second cas ce volume diminue après 17 secondes. En effet, à ce moment, l'eau atteint les limites du domaine et comme la condition aux limites est un bord ouvert, l'eau s'écoule en dehors du domaine.

A la fin des 60 secondes simulées, l'étendue de l'eau est beaucoup moins importante dans le cas d'un facteur d'atténuation égal à 0,6 avec 10% de surface mouillée contre 91%de surface dans le cas d'un facteur d'atténuation de 1,0. Cette différence est illustrée à la Figure 52.



a) b)

Figure 52 : Hauteur d'eau (m) après 60 secondes simulées dans le cas d'un facteur d'atténuation égal à 0,6 (a) 1,0 (b)

Dans la suite de ce travail, le facteur d'atténuation sera imposé à 1.

2.2 Facteur d'échelle

Le programme SARndbox possède un facteur d'échelle modifiable à l'exécution par l'utilisateur. En effet, dans la configuration par défaut, l'échelle entre le bac et le terrain modélisé est fixée à 1/100. Ce facteur d'échelle est appliqué sur l'étendue du bac, ce qui signifie que le bac représente une surface de 7,5 m par 10m, ce qui entraine donc une mise à l'échelle des tailles de cellules du domaine. De plus, cette échelle est aussi appliquée verticalement, toutes les valeurs entrées lors de l'exécution ou celles pas défaut qui concernent des grandeurs verticales comme l'intervalle de pluie, l'intensité de pluie mais aussi les valeurs captées par la Kinect.

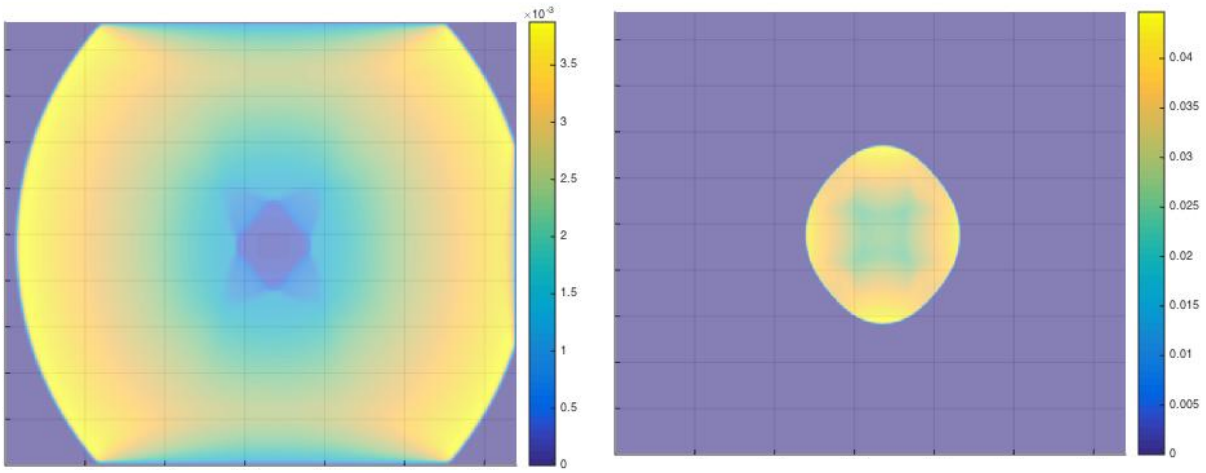
Le Tableau 14 présente la comparaison entre deux échelles au niveau des valeurs des paramètres utilisé lors du calcul d'écoulement.

	Échelle 1/1	Échelle 1/100
Taille de la texture de quantité	Selon x : 640 pixels Selon y : 480 pixels	Selon x : 640 pixels Selon y : 480 pixels
Taille des cellules (m)	Selon x : 0,00150353 Selon y : 0,00146828	Selon x : 0,150353 Selon y : 0,146828
Étendue modélisée (m)	Selon x : 0,96m Selon y : 0,70 m	Selon x : 96m Selon y : 70 m
Hauteur de 1cm dans le bac = m de l'étendue modélisée	0,01m	1m

Tableau 14: Comparaison de la valeur des variables en fonction du facteur d'échelle

Il est à noter que la bathymétrie et les quantités imposées par l'utilisateur dans cette étape d'analyse ne seront pas mises à l'échelle par le programme, seulement les données captées par la Kinect seront mises à l'échelle.

Si on conserve le cas de base défini au Tableau 13 en utilisant une échelle de 1/1 et de 1/100 on obtient après 60 secondes simulées en imposant la même quantité d'eau initiale les résultats de la Figure 53. On remarque bien que l'étendue d'eau est limitée dans le cas où la surface modélisée est plus grande.



a) b)

Figure 53: Hauteur d'eau (m) après 60 secondes simulées dans le cas d'une échelle égale à 1/1 (a) 1/100 (b)

2.3 Conditions aux limites

La condition aux limites établie dans le programme est une condition aux limites fortes de bords ouverts. Dans le calcul d'écoulement, la condition est implémentée après le calcul de l'intégration temporelle. Si lors de ce calcul il existe une hauteur d'eau sur les bords du domaine, elle est remise à 0 ainsi que les débits spécifiques. Cette condition induit donc que de l'eau peut sortir du domaine, la conservation de la masse d'eau n'est pas assurée. Comme cette condition n'a pas de sens physique, l'imposition d'une condition aux limites imperméables va être comparée à la condition initialement imposée. Cette imposition a consisté à augmenter la topographie sur le contour du domaine de manière à empêcher que l'eau puisse passer au-dessus.

La Figure 54 présente un graphe de l'évolution temporelle de la hauteur d'eau au centre du pixel à la position dans la texture (439,320) superposée à l'évolution du volume d'eau. Comme attendu le volume d'eau semble rester constant dans le cas de conditions aux limites imperméables alors qu'il diminue dans le cas de base. L'évolution du volume d'eau et de la hauteur d'eau au pixel définie reste identique jusqu'à 26 secondes. C'est à partir de ce moment que le front d'eau atteint un des bords du domaine et donc, dans le cas de base l'eau s'écoule en dehors du domaine. Il est alors normal que la hauteur d'eau décroisse progressivement puisque le domaine se vide. Avec les conditions imperméables puisque l'eau est arrêtée par la topographie importante aux bords, la hauteur augmente progressivement puisque l'écoulement d'eau continue de se propager vers les bords. Une fois que l'énergie cinétique de l 'écoulement incident est inférieure à l'énergie potentielle

aux bords alors, la direction de propagation de l'écoulement est inversée. Ce qui constitue la réflexion du front d'onde sur le bord.

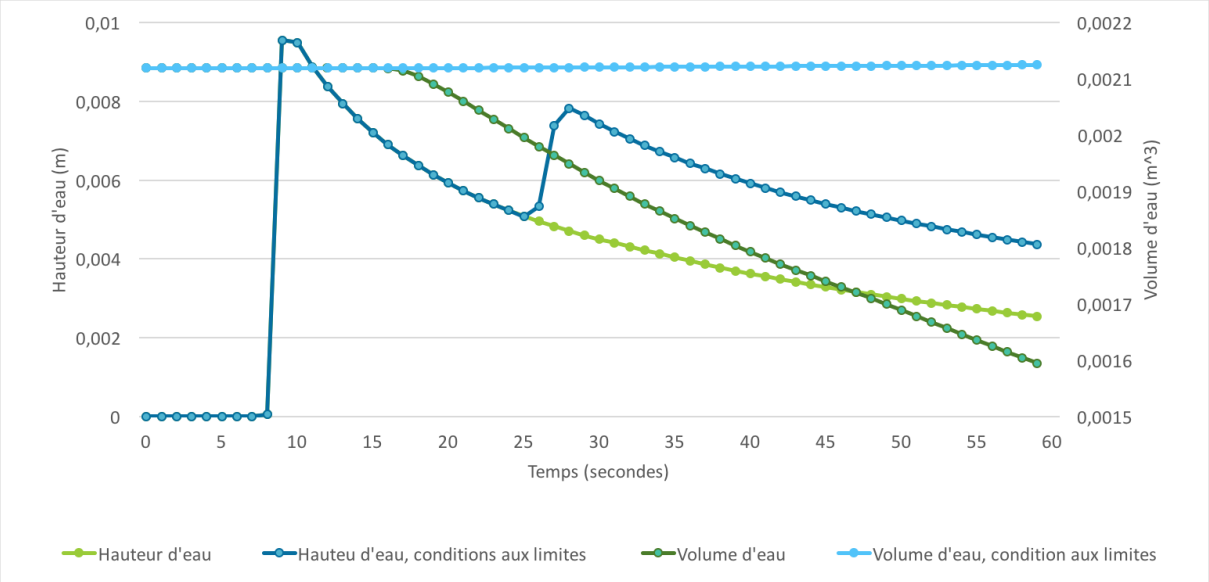
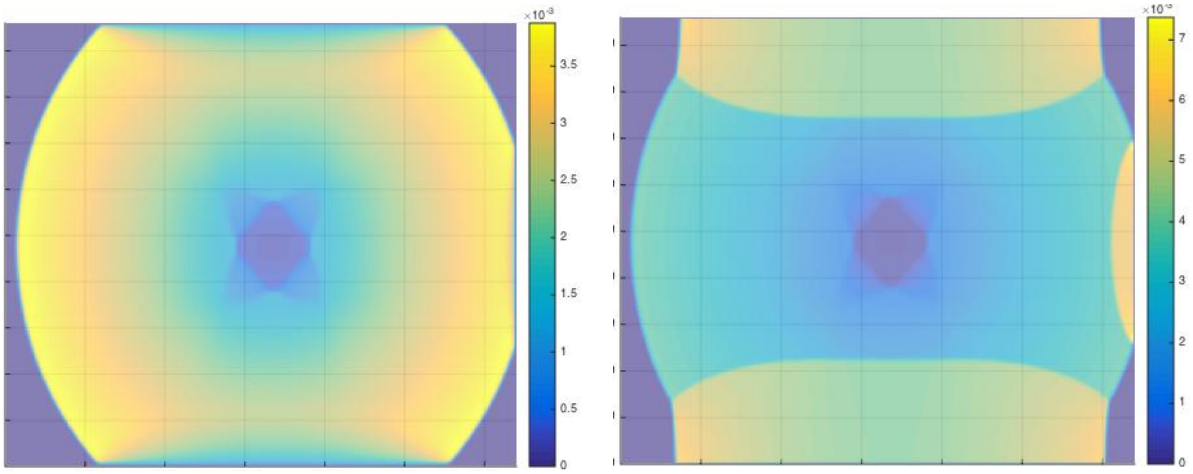


Figure 54: Evolution de la hauteur d'eau et du volume d'eau d'un pixel en fonction du temps pour deux types de conditions aux limites

La hauteur d'eau sur tout le domaine après 60 secondes, est présentée à la Figure 55. On remarque que la hauteur d'eau au centre d'un domaine est plus importante dans le cas de conditions imperméables. Dans ce cas-là, on observe bien l'onde de réflexion sur trois bords ce qui condense la masse d'eau au centre du domaine.



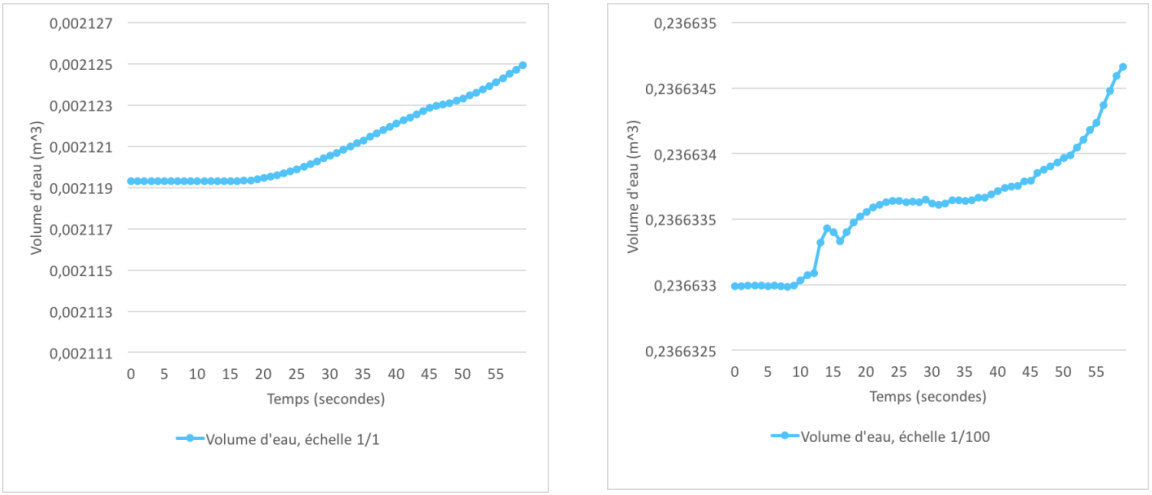
a) b)

Figure 55: Hauteur d'eau (m) après 60 secondes simulées dans le cas de base (a) de conditions aux limites imperméables (b)

2.4 Erreur détectée

A l'analyse numérique des résultats obtenus, lors du changement de la topographie avec des conditions pour que l'eau ne puisse s'écouler en dehors du domaine, il a été observé que la hauteur d'eau sur les cellules du périmètre du domaine étaient erronées. En effet, de l'eau était très faiblement générée (de l'ordre de 10^{-15} m) sur les 2 bords en y=1 y=489 et sur les cellules adjacentes.

Ceci a été mis en évidence avec les résultats calculés avec une topographie nulle et un facteur d'échelle 1/100 dû à la taille cent fois plus importantes des cellules. Dans les résultats à l'échelle 1/1, cela ne se manifeste que sur quelques cellules au droit des sommets du domaine. Cet apport d'eau non imposé et incohérent, provoque donc une augmentation de volume d'eau dans le modèle. Cette augmentation est visible à la Figure 56. Le premier cas correspond au cas de base avec des conditions aux limites pour que l'eau ne sorte pas du domaine calculé avec une échelle 1/1. Le second cas est le cas de base avec une échelle 1/100 et l'eau imposée égale à 10m sur les mêmes mailles que le cas de base.



a) b)

Figure 56: Évolution du volume d'eau en fonction du temps pour une échelle de 1/1 (a) et 1/100 (b)

Dans le premier cas de facteur d'échelle 1/1, l'augmentation de volume après 60 secondes est de 0,27% ce qui reste très faible. Dans le cas d'une échelle de 1/100, l'augmentation de volume après 60 secondes est de 0,0007%. On a donc une influence significative de l'échelle imposée. On peut noter que cette augmentation progressive apparaît lorsque que le front d'eau d'onde atteint une cellule du bord. Comme étudié dans la suite de ce chapitre, la topographie imposée influence aussi cette augmentation de volume.

L'investigation de la source de cette erreur a montré que les opérations de moyennage de la bathymétrie utilisée pour l'obtention de la hauteur d'eau au centre de la maille sont correctes. L'erreur peut venir du calcul des flux ou d'autres opérations numériques incorrectes et non identifiées. Dans le temps imparti pour ce travail, la recherche de la source de cette erreur et la correction n'a pu être poursuivie.

3. Comparaison au logiciel Wolf 2D

Bien qu'une erreur d'augmentation de volume persiste dans le code du programme SARnbox, la comparaison avec Wolf2D est envisagée pour déterminer si les résultats restent cohérents et l'allure des écoulements est sensiblement la même. Cette analyse permettra de donner un aperçu des performances du programme SARndbox auquel on pourrait s'attendre.

Wolf2D est un logiciel de calcul d'écoulement développé à l'université de Liège qui permet de réaliser des simulations numériques d'écoulement quasi en 3 dimensions en utilisant une discrétisation spatiale de type volumes finis ainsi qu'une intégration temporelle. Les simulations exécutées avec le logiciel se basent sur des équations des « shallow water » en prenant en compte une loi de frottement de Manning et la turbulence ainsi que des effets sédimentaires. [19]

Afin de comparer les résultats du programme SARnbox aux résultats du programme Wolf 2D, il a été défini deux topographies de référence :

- une surface plate
- une surface sous forme de double sinusoïde

Ces deux cas ont été simulés dans les deux logiciels en imposant le même nombre de courant, la même taille de maille ainsi que les mêmes conditions aux limites et initiales. Le Tableau 15 présente ces caractéristiques imposées aux deux programmes. Les deux programmes utilisent des schémas de discrétisation en volumes finis mais une implémentation différente comme expliqué précédemment.

Caractéristiques	Valeurs
Conditions aux limites aux bords	- Hauteur d'eau nulle - Débits spécifiques selon x et y nuls
Critère de Courant	0,5
Taille de la texture de quantité	Selon x : 640 pixels Selon y : 480 pixels
Taille des cellules (m)	Selon x : 0,00150353 Selon y : 0,00146828
Échelle	1 /1

Tableau 15:Caractéristique des cas de références

Les résultats du programme WOLF2D pour cette comparaison nous ont été fournis. Les deux programmes n'étant pas utilisés sur un même ordinateur, les performances en temps de calcul présentés sont fonction de la configuration de l'ordinateur utilisé. La configuration d'ordinateur utilisée pour le programme Wolf2D est disponible à l'Annexe 8.

Le schéma d'intégration temporelle utilisé dans le logiciel Wolf est un schéma Runge-Kutta d'ordre 3 qui permet un temps de calcul réduit parce que le pas de temps est plus important.

3.1 Topographie nulle

Le premier cas étudié est une topographie de hauteur constante et nulle partout avec une imposition initiale d'une hauteur d'eau uniforme de 10 cm sur une portion du domaine et des débits spécifiques

nuls. Cette condition initiale est illustrée à la Figure 57. L'écoulement a été simulé pendant 60 secondes avec une sauvegarde des résultats toutes les secondes.

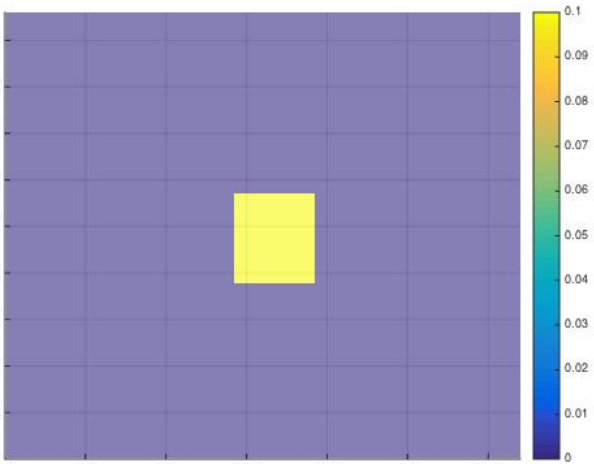
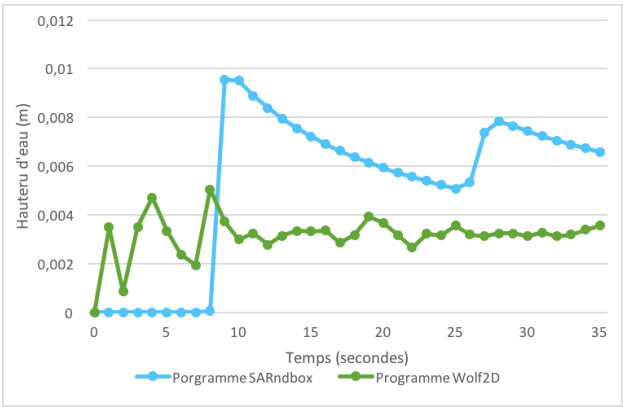


Figure 57: Hauteur d'eau initiale imposée

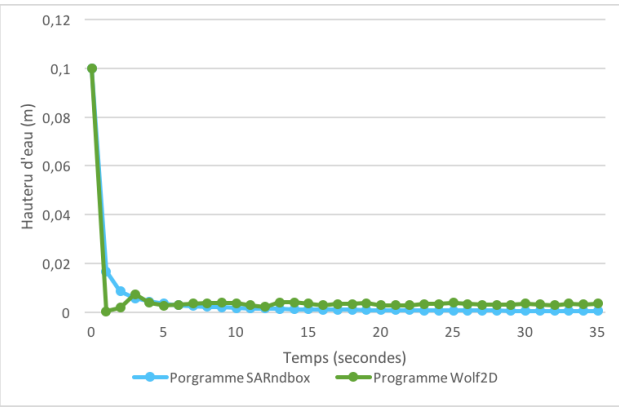
La Figure 58 présente la comparaison de l'évolution temporelle de 4 points du domaine :

- sur le périmètre du domaine : pixel (439,320)
- dans la surface où initialement est imposée de l'eau: pixel (250,300)
- dans la surface où initialement est imposée de l'eau: pixel (100,50).

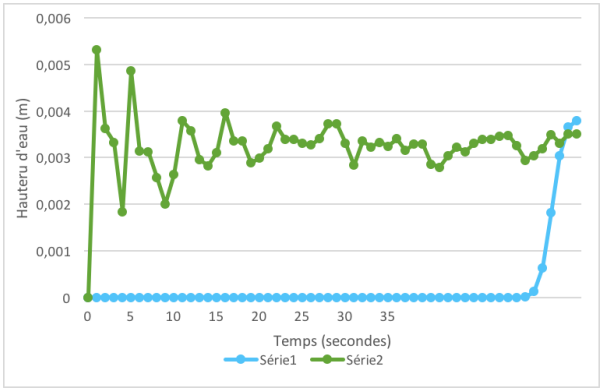
On remarque que les courbes de hauteur d'eau ne se ressemblent pas du tout. On remarque qu'après 1 seconde de simulation les 10 centimètres d'eau imposée sur une portion de la surface initiale se sont répartis sur presque tout le domaine avec le programme Wolf2D alors que dans le programme SARndbox, l'eau n'est répartie que sur une faible surface. Cette incohérence est visible aussi à la Figure 59 montrant la hauteur d'eau sur tout le domainereprésenté aux temps 1, 5,20 et 60 secondes.



a) Pixel (439,320)

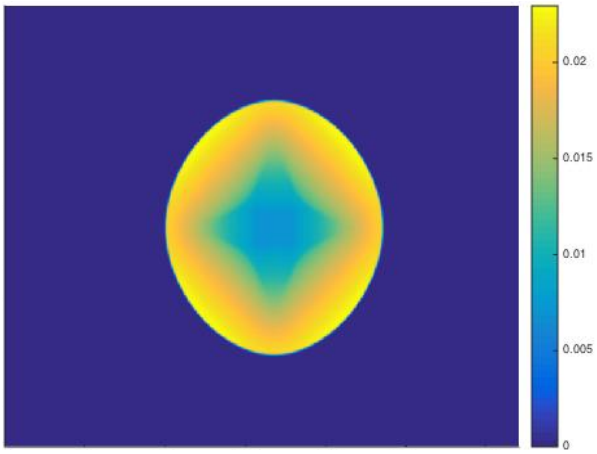


b) Pixel (250,300)

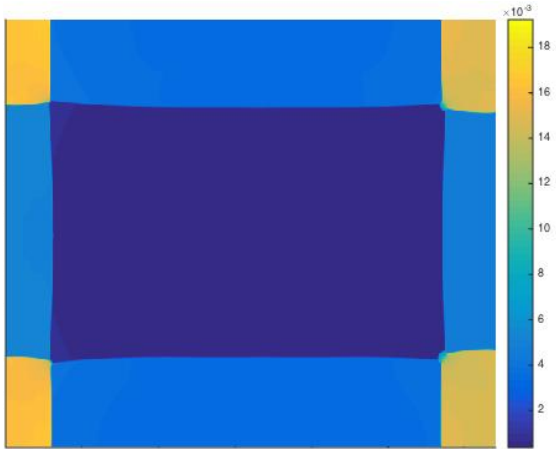


c) Pixel (100,50)

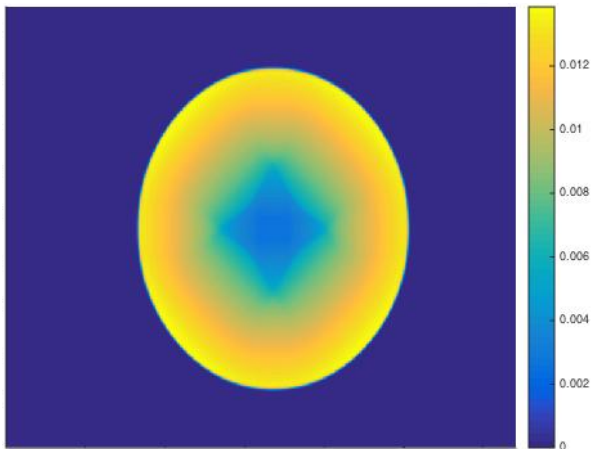
Figure 58: Évolution temporelle de la hauteur d'eau (en m) de 4 points du domaine



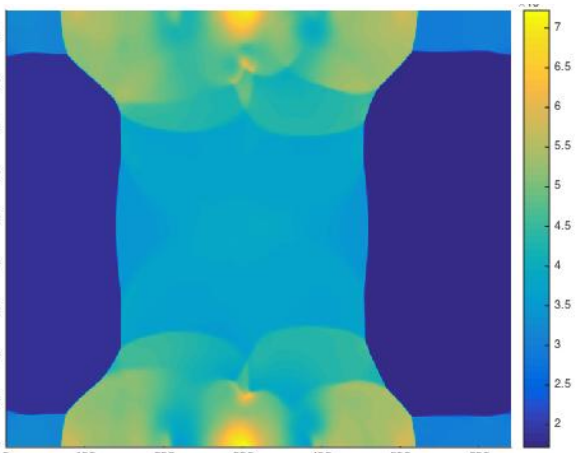
a) SARnbox : Temps=1s



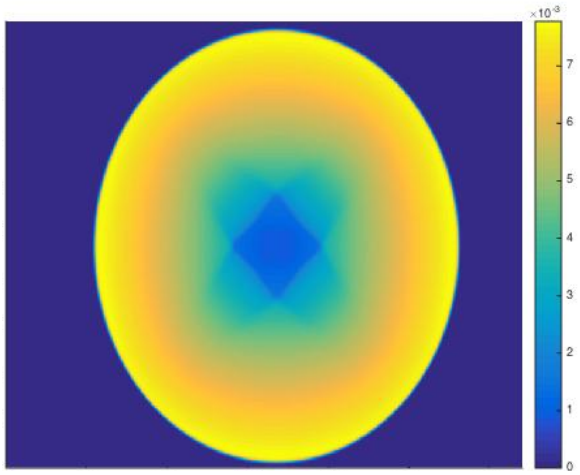
b) Wolf2D : Temps=1s



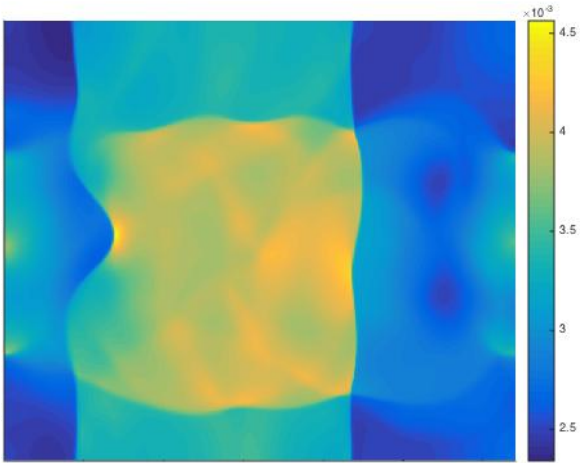
c) SARnbox : Temps=5s



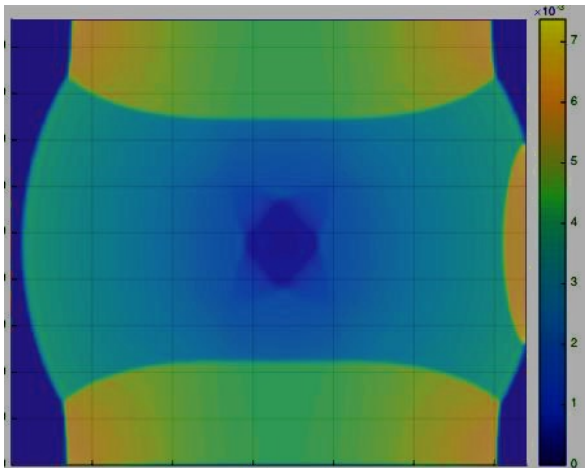
d) Wolf2D : Temps=5s



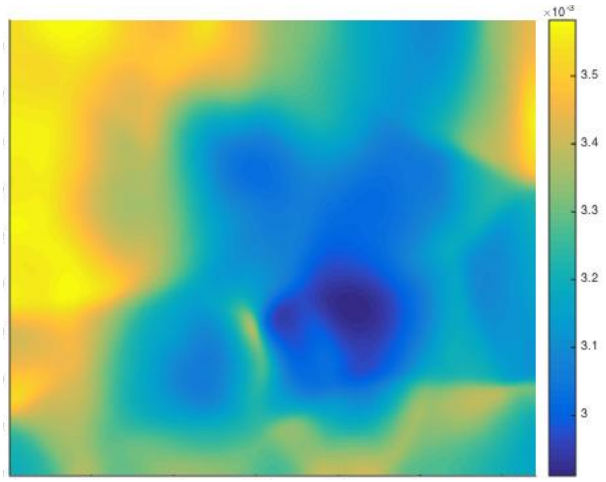
e) ARnbox : Temps=20s



f) Wolf2D: Temps=20s



g) SARnbox : Temps=60s



h) Wolf2D : Temps=60s

Figure 59: Hauteur d'eau (m) sur tout le domaine en temps discrets

Au vu de l'allure des courbes, il se pourrait que les résultats obtenus avec un des deux programmes soit erronés. En effet, même après une seconde les résultats simulés sont totalement différents. Des autres simulations à partir de cette topographie seraient intéressantes à mener pour vérifier ou non cette divergence.

Le Tableau 17 présente les temps de calculs pour effectuer cette simulation de 60 secondes dans chaque programme.

	SARndbox	Wolf2D
Temps simulé (s)	60	
Temps de calcul (s)	10	60
Nombre de pas de temps	21744	12277
Pas de temps moyen (s)	0,0027594	0,005

Tableau 16 : Performances de calcul du programme SARndbox et Wolf2D

On remarque que le temps d'exécution est beaucoup plus faible dans le cas du programme SARndbox alors que le nombre de pas de temps est beaucoup plus important que dans le cas du programme Wolf2D.

3.2 Topographie en double sinusoïde

Le cas d'une double sinusoïde symétrique dans les 2 dimensions a été envisagée afin de regarder si une quantité d'eau initiale imposée sur tout le domaine se réparti dans les creux des sinusoïdes et se stabilise après un certain temps. La bathymétrie imposée est disponible à la Figure 60.

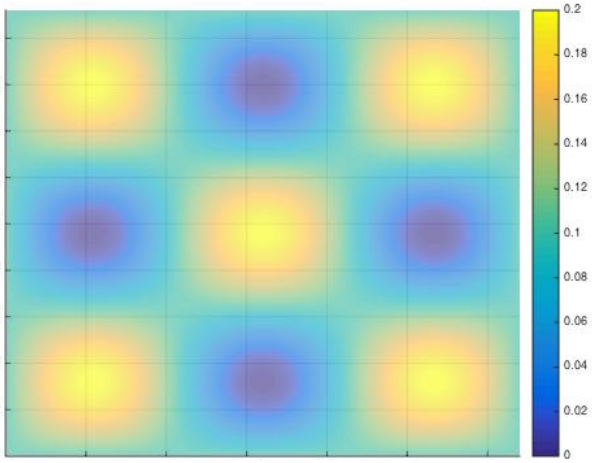


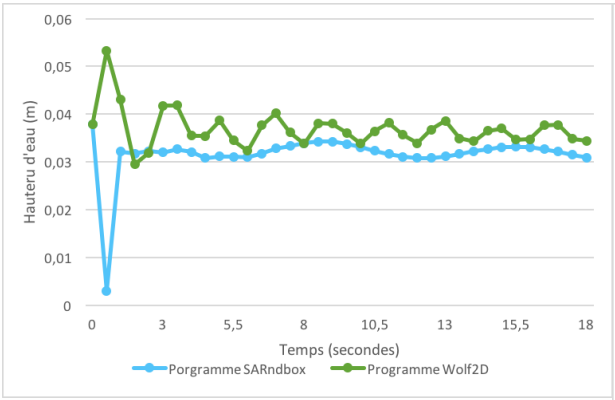
Figure 60: Topographie en double sinusoïde imposée

La quantité d'eau initiale a été déterminée en calculant le volume des creux de la sinusoïde et en le répartissant sur tout le domaine uniformément soit une hauteur d'eau uniforme de 0,038 m et les débits spécifiques initiaux nuls.

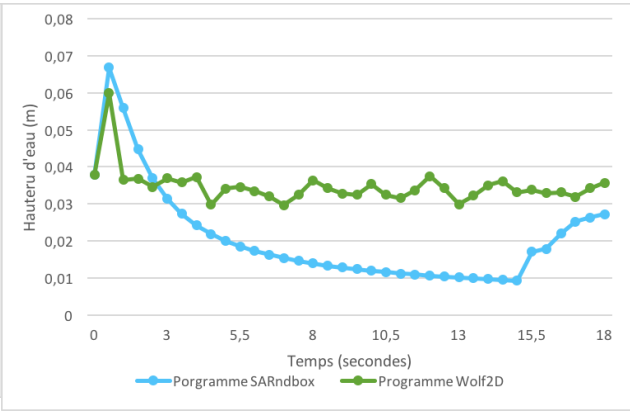
La Figure 61 présente la comparaison de l'évolution temporelle de 4 points du domaine :

- sur le périmètre du domaine : pixel (639,240)
- en un coin du domaine : pixel (2,2)
- dans un creux de la surface : pixel (100,250)
- sur un pic de la surface : pixel (500,400)

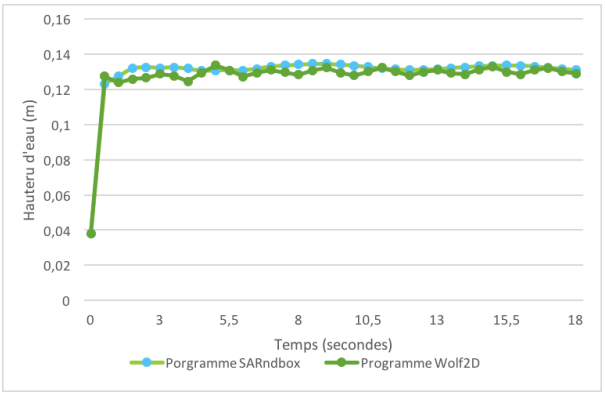
On remarque que l'allure des courbes est sensiblement la même avec des écarts de valeurs de +/- 1cm. Cependant, on remarquera qu'au coin du domaine, l'allure des deux courbes n'est pas la même, cela peut laisser penser que l'eau n'est pas réfléchie en ce bord. La hauteur d'eau persistante au droit d'un pic de topographie dans le SARndbox constitue aussi un élément non cohérent des résultats obtenus. Ces deux différences majeures peuvent êtres dues à l'erreur détectée qui, dans ce cas, entraine une augmentation de volume non négligeable de 11% après 30 secondes.



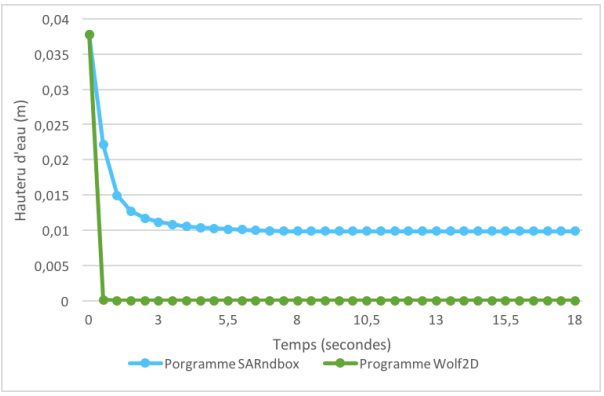
d) Pixel (639,240)



e) Pixel (2,2)



f) Pixel (100,250)



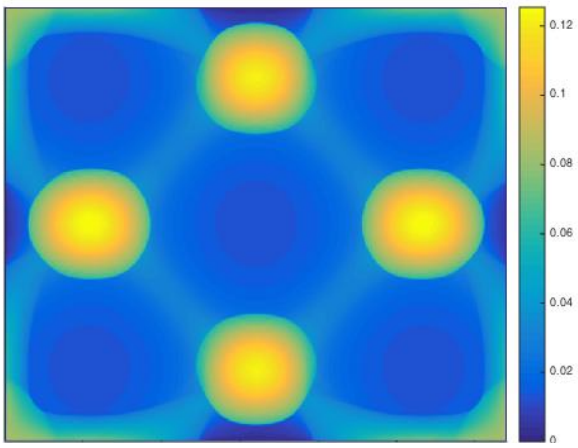
g) Pixel (500,400)

Figure 61: Évolution temporelle de la hauteur d'eau (en m) de 4 points du domaine

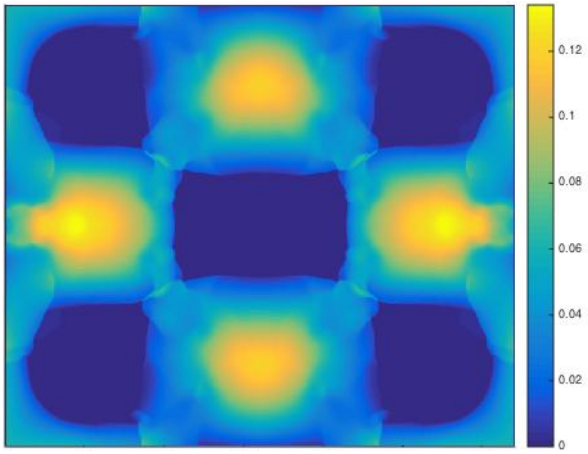
La Figure 62présente la hauteur d'eau sur tout le domaine en temps discret :

- 0,5s
- 2,5s
- 7,5s
- 15s

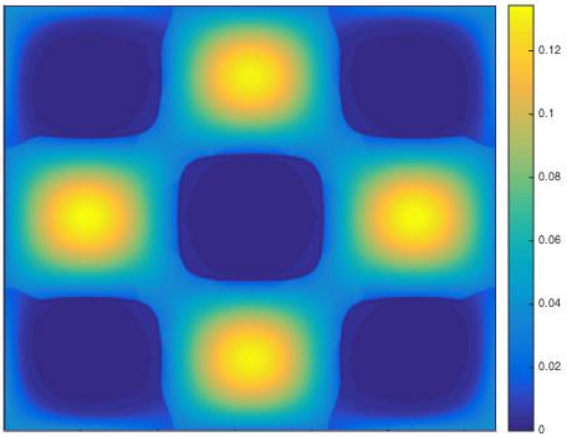
On peut noter que les aspects de la répartition spatiale en ces 4 temps sont sensiblement les mêmes malgré quelques différences aux bords du domaine et au début de la simulation. Dans, les premières secondes, on distingue une répartition de l'eau initialement imposée sur tout le domaine. Ensuite, l'eau se stabilise dans les creux de la surface après au moins 7,5 secondes.



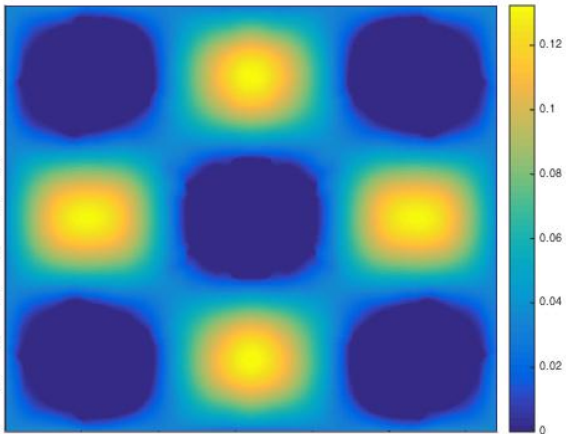
i) SARnbox : Temps=0,5s



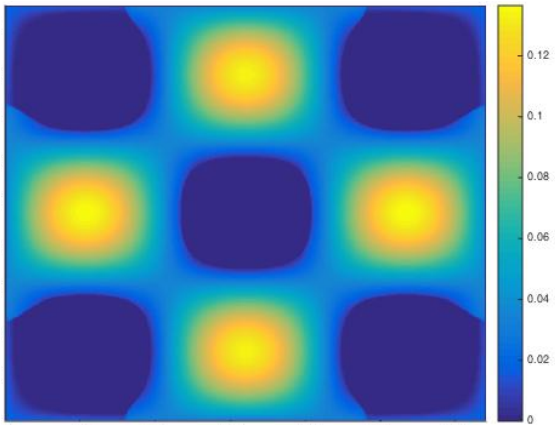
j) Wolf2D : Temps=0,5s



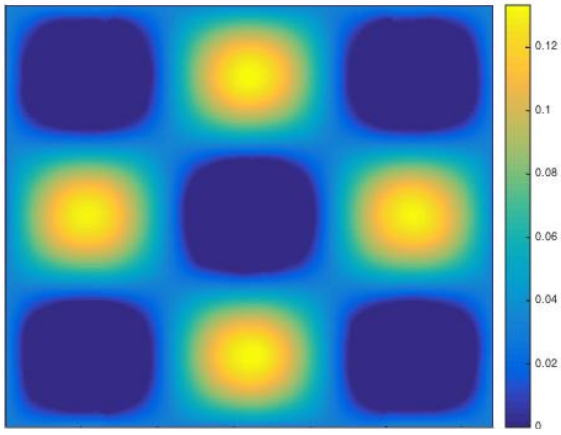
k) SARnbox : Temps=2,5s



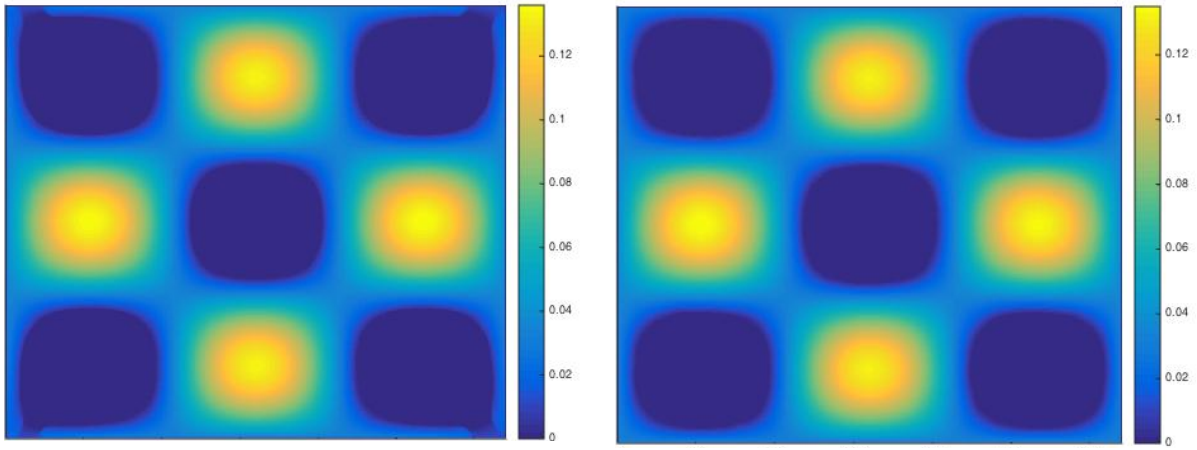
l) Wolf2D : Temps=2,5s



m) SARnbox : Temps=7,5s



n) Wolf2D: Temps=7,5s



o) SARnbox : Temps=15s p) Wolf2D : Temps=15s

Figure 62: Évolution temporelle de la hauteur d'eau (en m) de 4 temps discret

Le Tableau 17 présente les temps de calculs pour effectuer cette simulation de 60 secondes dans chaque programme.

	SARndbox	Wolf2D
Temps simulé (s)	30	
Temps de calcul	43 s	60 s
Nombre de pas de temps	80157	12094
Pas de temps moyen (s)	0,0074883	0,0025

Tableau 17 : Performances de calcul du programme SARndbox et Wolf2D

On remarque que le temps d'exécution est plus faible dans le cas du programme SARndbox que dans le cas du programme Wolf2D alors que l'algorithme à 3 pas est plus rapide.

3.3 Résultats de la comparaison

A travers les 2 cas de topographie imposées aux programmes, même si l'allure des courbes est similaire, l'analyse des résultats numériques montre des disparités.

Dans le cas d'un topographie plane, ces écarts sont très importants et suggèrent qu'il demeure des erreurs dans le programme SARndox malgré les corrections déjà effectuées. Les divergences observées devraient être vérifiées grâce à d'autre cas de condition initiale avec cette topographie.

Sur le cas de topographie en double sinusoïde, ces écarts sont moins importants

Une comparaison des programmes basée sur un seul cas de surface imposée est peu représentatif. On remarque une évolution temporelle de la hauteur d'eau semblable avec les deux programmes et des incohérence sur les pixels du bord, probablement liées à l'erreur détectée au Chapitre 6 -2.4 . Au niveau de l'évolution temporelle de certains pixels, les valeurs de deux programmes sont proches avec une différence de valeur +/- de 1cm.

Malgré ces résultats non cohérents, l'avantage de la résolution numérique à l'aide des GSLS shaders sur GPU apparaît clairement en un temps de résolution plus faible pour un pas de temps plus petit.

Il serait hasardeux de tirer des conclusions à partir d'un seul cas de topographie imposée et sans vérifier la présence d'erreurs à partir d'autre scénarios. A ce stade aucune tendance de comportement entre les 2 programmes ne peut être présentée.

Table des figures

Bibliographie

[1] R. Silva, J. C. Oliveira en G. A. Giraldi, „Introduction to augmented reality," *National laboratory for scientific computation,* 2003.

[2] T. Ali, „FA-18TRB," 20 mai 2010. [Online]. Available: http://talha-ali.blogspot.be/2010/05/fa-18trb.html. [Geopend 5 avril 2017].

[3] N. Navab, J. Traub, T. Sielhorst, M. Feuerstein, en C. Bichlmeier, „Action-and workflow-driven augmented reality for computer-aided medical procedures," *IEEE Computer Graphics and Applications,* vol. 27, nr. 5, pp. 10--14, 2007.

[4] J.-S. Kriegk, „Réalité augmentée: la révolution du jeu vidéo est en marche à Bordeaux, chez Asobo," HuffPost, 28 novembre 2016. [Online]. Available: http://www.huffingtonpost.fr/jeansamuel-kriegk/realite-augmentee-la-revolution-du-jeu-video_b_9678380.html. [Geopend (avril 2017].

[5] M. R. Andersen, T. Jensen, P. Lisouski, A. K. Mortensen, M. K. Hansen, T. Gregersen en P. Ahrendt, „Kinect depth sensor evaluation for computer vision applications," *Electrical and Computer Engineering Technical Report ECE-TR-6,* 2012.

[6] S. Chourasiya, P. K. Mohapatra en S. Tripathi, „Non-intrusive underwater measurement of mobile bottom surface," *Advances in Water Resources,* vol. 104, pp. 76--88, 2017.

[7] B. Lange, R. Juang, E. Suma, M. Bolas en A. Rizzo , „Interactive game-based rehabilitation using the Microsoft Kinect," *Virtual Reality Short Papers and Posters (VRW), 2012 IEEE,* pp. 171--172, 2012.

[8] K. D. Mankoff en T. A. Russo, „The Kinect: A low-cost, high-resolution, short-range 3D camera," *The Kinect: A low-cost, high-resolution, short-range 3D camera,* vol. 38, pp. 926--936, 2013.

[9] A. Lejeune, S. Piérard, M. Van Droogenbroeck en J. Verly, „Utilisation de la Kinect," *Linux Magazine France,* vol. 151, pp. 16--29, 2012.

[10] S. Zennaro, M. Munaro, S. Milani, P. Zanuttigh, A. Bernardi, S. Ghidoni en E. Menegatti, „erformance evaluation of the 1st and 2nd generation Kinect for multimedia applications," *Multimedia and Expo (ICME), 2015 IEEE International Conference on,* pp. 1--6, 2015.

[11] K. Khoshelham, „Accuracy analysis of kinect depth data," *ISPRS workshop laser scanning,* vol. 38, p. 5, 2011.

[12] D. Pagliari en L. Pinto, „Calibration of kinect for xbox one and comparison between the two generations of Microsoft sensors," *Sensors,* vol. 15, pp. 27569--27589, 2015.

[13] O. Kreylos, „Research and Development Homepage," [Online]. Available: http://idav.ucdavis.edu/~okreylos/ResDev/SARndbox/index.html. [Geopend 11 août 2016].

[14] Boouh, „ Développez vos applications 3D avec OpenGL 3.3," OpenClassRooms, 13 mars 2017. [Online]. Available: https://openclassrooms.com/courses/developpez-vos-applications-3d-avec-opengl-3-3/les-frame-buffer-objects. [Geopend 25 avril 2017].

[15] A. Laurent, „Les shaders dans OpenGL," Devellopez.com, 24 mai 2011. [Online]. Available: http://alexandre-laurent.developpez.com/tutoriels/OpenGL/OpenGL-GLSL/?page=page_3. [Geopend 29 avril 2017].

[16] M. Nebra en S. Matthieu, „Programmez en langage C++: Les classes," Open Classrooms, 2 mai 2017. [Online]. Available: https://openclassrooms.com/courses/programmez-avec-le-langage-c/introduction-la-verite-sur-les-strings-enfin-devoilee. [Geopend 3 mai 2017].

[17] A. Kurganov en G. Petrova, „A second-order well-balanced positivity preserving central-upwind scheme for the Saint-Venant system," *Communications in Mathematical Scienc,* vol. 5, pp. 133--160, 2007.

[18] B. Allegretto, „THE TAHOE DAILY SNOW," [Online]. Available: https://opensnow.com/dailysnow/tahoe/post/5583.

[19] „La suite logicielle WOLF," [Online]. Available: http://www.hach.ulg.ac.be/cms/node/5.

[20] „Augmented reality sandbox," 2016. [Online]. Available: https://arsandbox.ucdavis.edu. [Geopend 30 septembre 2016].

[21] L. Yang, L. Zhang, H. Dong, A. Alelaiwi en A. El Saddik, „Evaluating and improving the depth accuracy of Kinect for Windows v2," *IEEE Sensors Journal,* vol. 15, pp. 4275--4285, 2015.

Annexe 1 : Paramètres internes des modèles Kinect

	Nom de la caméra			
	Kinect 1.0 RGB Camera		Kinect 1.0 IR Camera	
	Imaging Sensor			
Type	Aptima MT9M112 CMOS		Aptima MT9M001 CMOS	
Résolution(pixels)	1280x1024 ou 640x480		640x480	
Taille d'un pixel (µm)	2,8		5,2	
	Paramètres internes			
	Valeurs	**Ecart-type**	**Valeurs**	**Ecart-type**
Longueur focale (mm)	3,099	2,03 e^{-3}	6,97	3,0 e^{-3}
Largeur de l'image (pixel)	3,58		6,66	
Hauteur de l'image (pixel)	2,87		5,32	
Largeur de l'image (pixel)	640		640	
Hauteur de l'image (pixel)	480		480	
Point principal x(mm)	-0,040	9,2 e^{-4}	-0,005	2,0 e^{-3}
Point principal y(mm)	-0,020	1.0 e^{-3}	-0,004	3,0 e^{-3}
	Paramètres additionnels			
$K_1 (mm^{-2})$	-1,366 e^{-3}	9,1 e^{-5}	1,759 e^{-3}	4,3 e^{-5}
$K_2 (mm^{-4})$	7,857 e^{-4}	1,7 e^{-5}	-8,337 e^{-5}	2,5 e^{-6}
$P_1 (mm^{-2})$	-1,518 e^{-4}	2,9 e^{-5}	-1,835 e^{-4}	2,1 e^{-5}
$P_2 (mm^{-2})$	-9,514 e^{-4}	3,2 e^{-5}	2,538 e^{-4}	2,2 e^{-5}

Tableau 18: Paramètres interne des caméras de la Kinect v1 durant la phase de calibration [12]

	Nom de la caméra			
	Kinect 2.0 RGB Camera		Kinect 2.0 IR Camera	
	Imaging Sensor			
Type	-		-	
Résolution(pixels)	1920x1080		512x424	
Taille d'un pixel (µm)	3,1		10	
	Paramètres internes			
	Valeurs	**Ecart-type**	**Valeurs**	**Ecart-type**
Longueur focale (mm)	3,291	1,0 e^{-3}	3,657	5,2 e^{-4}
Largeur du format (mm)	6,00		5,12	
Hauteur du format (mm)	3,38		4,24	
Largeur de l'image (pixel)	1920		512	
Hauteur de l'image (pixel)	1080		424	
Point principal x(mm)	-0,005	5,6 e^{-4}	0,032	3,5 e^{-4}
Point principal y(mm)	-0,016	6,9 e^{-4}	0,033	3,9 e^{-4}
	Paramètres additionnels			
$K_1 (mm^{-2})$	3,823 e^{-3}	3,8 e^{-5}	-6,510 e^{-3}	2,7 e^{-5}
$K_2 (mm^{-4})$	3,149 e^{-4}	3,8 e^{-6}	1,205 e^{-3}	3,8 e^{-6}
$P_1 (mm^{-2})$	2,332 e^{-4}	2,0 e^{-5}	1,377 e^{-4}	8,0 e^{-6}
$P_2 (mm^{-2})$	-5,152 e^{-4}	2,1 e^{-6}	1,589 e^{-4}	9,2 e^{-6}

Tableau 19 : Paramètres interne des caméras de la Kinect v2 durant la phase de calibration [12]

Annexe 2 : Mises à jour du programme SARndbox

Version	Date de sortie	Améliorations et changements
1.0	/	• Version initiale
1.1	14/11/12	• Ajout du taux constant d'évaporation d'eau
1.2		• Changement des types des variables de taille en unsigned int
1.3		• Correction dans le code de calibration du projecteur (normalisation de l'homographie de calibration, calcul de la valeur projetée axe-z pour la valeur dans le buffer) • Gestion les vues non orthogonales de la Kinect
1.4		• Correction d'une erreur dans l'étape 5 des instructions d'installation dans le fichier README • Mise à jour des instructions d'installations présentes dans le fichier README pour l'explication du nouvel outil de calibration
1.5	16/03/2015	• Création de goutte non-cible en jaune pour les distinguer des gouttes cibles dans l'opération de calibration.
1.5-001		• Résolution d'une anomalie à l'origine de lignes bleues sur le pourtour de la surface de simulation • Résolution d'une anomalie qui empêche l'eau de « couler » en dehors du bac • Ajout de l'option d'hystérésis utilisé pour reconstruire les surfaces.
1.6		• Ajout d'une classe pour mettre en pause la captation de topographie • Ajout d'une fenêtre de contrôle dans l'application pour gérer les paramètres de simulation d'écoulements • Mise à jour du fichier README (lien vers une vidéo d'installation, étape de calibration • Ajout de commandes pour modifier les paramètres par défaut lors de l'exécution de l'application
2.0		• Restructuration complète du logiciel • Ajout de la fonctionnalité de « faire pleuvoir » sur base de la détection de main • Ajout d'un outil qui utilise un DEM. • Changement de certaines lignes de commande concernant les options d'exécution du programme. • Ajout d'une option pour imposer le facteur d'échelle
2.1		• Support des écrans multiples avec des paramètres de rendu individuel • Amélioration de l'outil DEM
2.2	16/10/2016	• Ajout d'un outil pour sauver la surface captée sous un format .dem
2.3	01/11/2016	• Résolution des erreurs de mise à l'échelle • Modification des intervalles par défaut pour l'élévation et la pluies

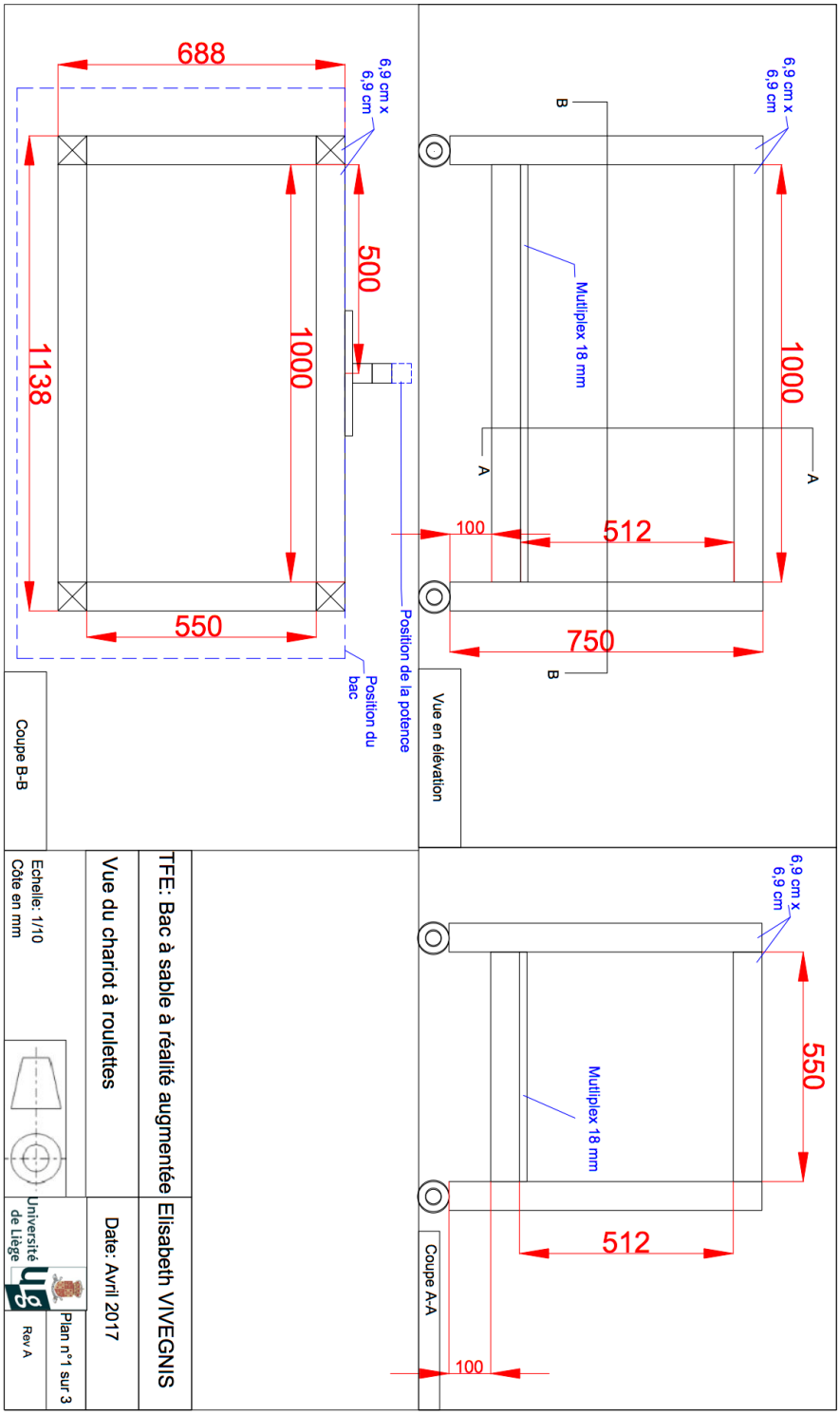
Tableau 20: Mise à jour du logiciel SARndbox [13]

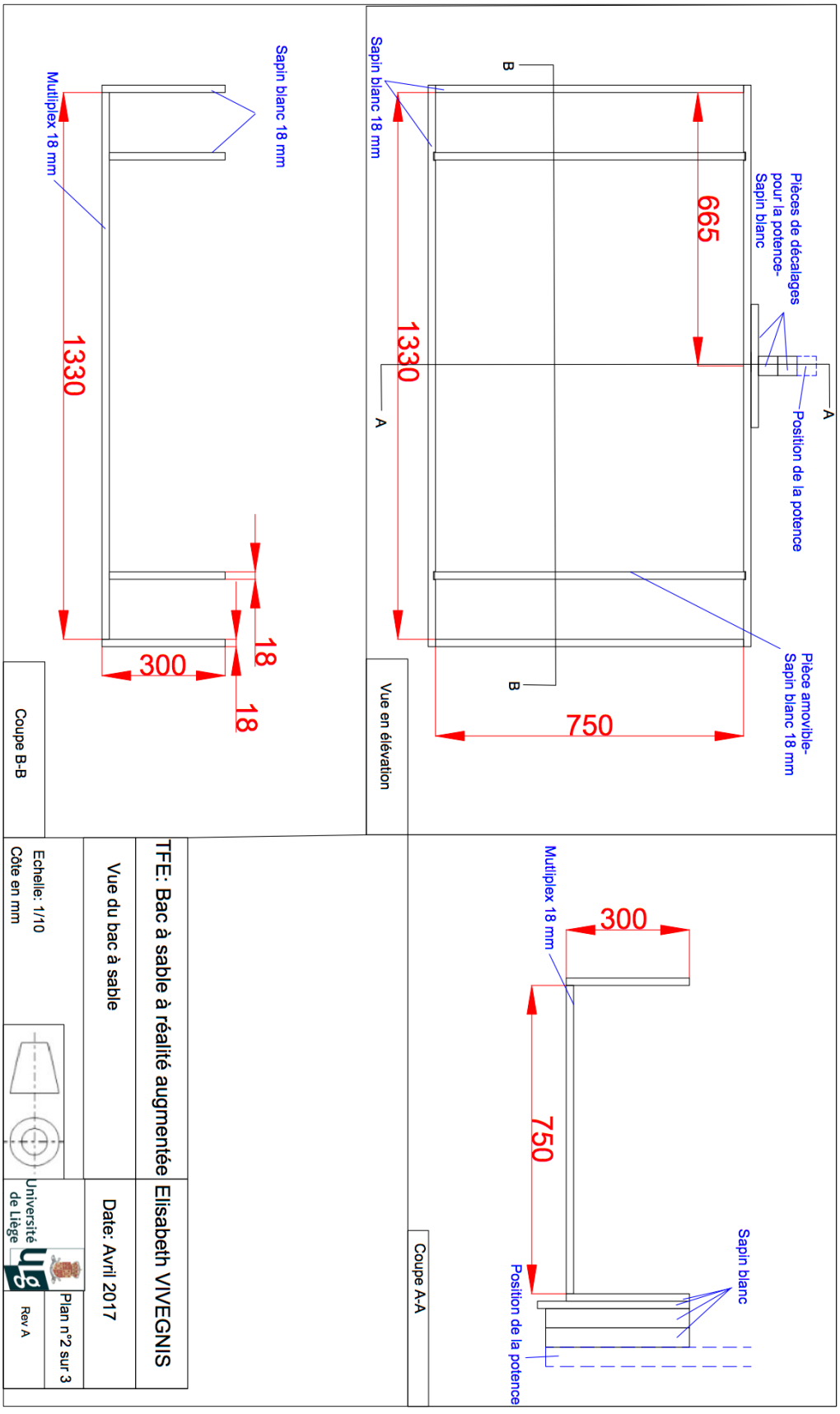
Annexe 3 : Paramètres du projecteur Optoma GT1080E

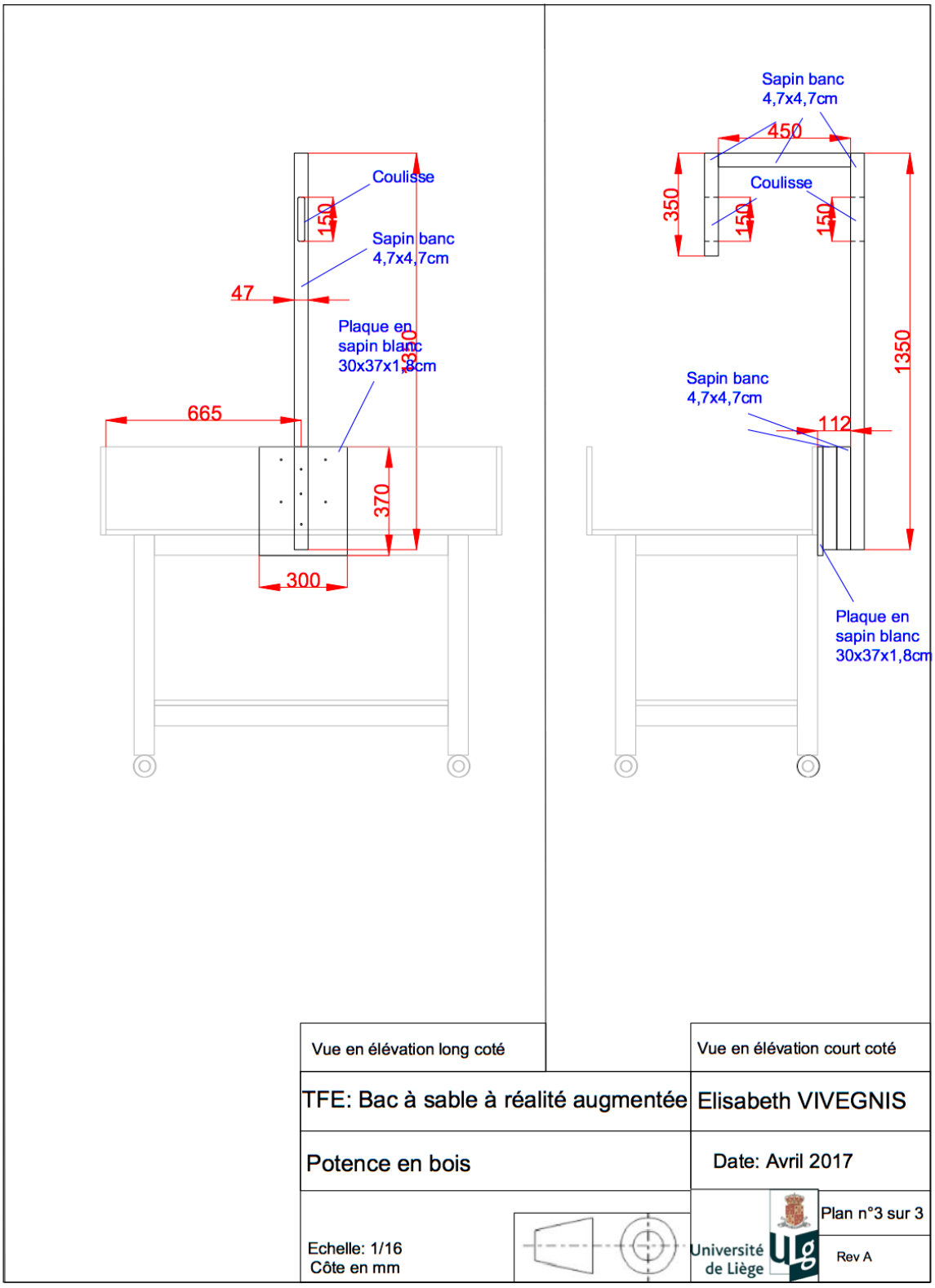
Specification

Technologie d'affichage	Technologie 0.65" DarkChip 3 1080p DLP® par Texas Instruments
Résolution	1080p 1920 x 1080
Luminosité [1] (mode Lumineux)	3000 ANSI Lumens
Contraste	25 000:1
Lamp Life[2] Dynamic/Eco/Bright	6500/6000/5000 (hrs)
Rapport de projection	0.49:1
Type de zoom	Fixe
Connecteurs (Entrées/Sorties)	2 x HDMI (1.4a 3D support) + MHL v1.2, Audio Out 3.5mm, Relais 12V, 3D-Sync, USB service
Haut-parleur (Watts)	10
Correction Trapézoïdale	± 40° Vertical
Poids (kg)	2.65
Dimensions (LxPxH) (mm)	315 x 224 x 114
Ratio	16:9 Natif, Compatible 4:3
Offset	116% ±5%
Taille image projetée	1.15 - 7.67m (45.3" - 300") Diagonale 16:9
Distance de projection	0.5 - 3.35 mètres
Optique	F/2.8; f=7.42mm
Uniformité	80%
Résolution Maximum	Full HD 1920 x 1080
Compatibilité Informatique	UXGA, SXGA, WXGA, HD, XGA, SVGA, VGA, Mac
Compatibilité	PAL (B, D, G, H, I, M, N, 576i/p), NTSC (M, 4.43/3.58 MHz, 480i/p), SECAM (B, D, G, K, K1, L) HD (1080i, 720p)
Support 3D	Full 3D - Les fonctionnalités 3D des projecteurs Optoma peuvent être utilisées seulement avec des contenus 3D compatibles. Les applications typiques telles que les logiciels d'éducation 3D, de modélisation et de Design 3D. Les systèmes Broadcast TV 3D (SKY au RU), les Blu-ray 3D™ et les jeux 3D de la PS3 Sony® ou la Xbox 360 de Microsoft® sont maintenant supportés par les spécifications de la HDMI v1.4a
Compatibilité 3D	Side-by-Side:1080i50 / 60, 720p50 / 60 Frame-pack: 1080p24, 720p50 / 60 Over-Under: 1080p24, 720p50 / 60
Input Lag (ms)	33
Taux de rafraichissement Horizontal	15.375~91.146 KHz
Taux de rafraichissement Vertical	24 ~ 85Hz (120Hz pour la 3D)
Nombre de Couleurs	1073.4 Millions
Niveau sonore (mode Eco)	26dB
Alimentation	100-240V, 50-60Hz
Consommation électrique	233W mode Lumineux/ 193W mode Eco (< 0.5W lode veille)
Type de lampe	190W
Température de fonctionnement	5°C - 40°C, Max. Humidité 85%, Max. Altitude 3000m
Sécurité	Barre de sécurité, encoche Kensington, protection par mot de passe
Menu à l'écran	27 Langues: Français, Allemand, Anglais, Espagnol, Italien, Portugais, Hollandais, Suédois, Finlandais, Grec, Danois, Norvégien, Polonais, Russe, Chinois, Chinois traditionnel, Coréen, Arabe, Japonais, Thailandais, Hongrois, Tchèque, Turque, Vietnamiense, Farsi, Roumain, Indonésien
Accessoires fournis en standard	Cache de l'objectif, Sacoche, Câble HDMI, Câble d'alimentation, Télécommande avec piles, Carte de démarrage rapide, Manuel utilisateur sur CD,
Accessoires en option	Système 3D sans fil, Lunettes 3D sans fil, Lunettes 3D DLP® Link™, sans fil, fixation plafond
Sans fil (optionnel)	Yes
Garantie	2 Years
RoHS	Conforme

Annexe 4 : Plans du bac à sable




Sapin blanc 18 mm
Mutliplex 18 mm
1330
300
18
18
Coupe B-B
Sapin blanc 18 mm
1330
665
750
Pièces de décalages pour la potence- Sapin blanc
Position de la potence
Pièce amovible- Sapin blanc 18 mm
Vue en élévation
Mutliplex 18 mm
300
750
Sapin blanc
Position de la potence
Coupe A-A
TFE: Bac à sable à réalité augmentée
Elisabeth VIVEGNIS
Vue du bac à sable
Date: Avril 2017
Echelle: 1/10
Côte en mm
Université de Liège
Plan n°2 sur 3
Rev A


Coulisse
150
Sapin banc
4,7x4,7cm
47
Plaque en
sapin blanc
30x37x1,8cm
665
370
300
Sapin banc
4,7x4,7cm
450
Coulisse
350
150
150
1350
Sapin banc
4,7x4,7cm
112
Plaque en
sapin blanc
30x37x1,8cm
Vue en élévation long coté
Vue en élévation court coté
TFE: Bac à sable à réalité augmentée
Elisabeth VIVEGNIS
Potence en bois
Date: Avril 2017
Echelle: 1/16
Côte en mm
Université
de Liège
Plan n°3 sur 3
Rev A

Annexe 5 : Assemblage faux-tenon et mortaise

Annexe 6 : Procédure d'installation du logiciel SARndbox [20]

La procédure d'installation présentée constitue une traduction française de la procédure officielle avec quelques précisions dans le fonctionnement. [20] Deux vidéos d'installation sont disponibles sur les 2 liens suivant :

https://www.youtube.com/watch?v=R0UyMeJ2pYc&feature=youtu.be (étape 1 à 7)

https://www.youtube.com/watch?v=EW2PtRsQQr0 (étape 8 à 11)

1) Installation d'une version récente de Linux Mint 64 bits version MATE sur un ordinateur vierge.

2) Installation des drivers de la carte graphique Nvidia dans l'application « Administration Center » » sélectionner "Gestionnaire des pilotes" dans la liste des drivers disponibles, choisir le driver recommandé, sélectionner « Appliquer les changements ». Ensuite attendre que le changement se fasse et puis redémarrer l'ordinateur.

3) Ouvrir un terminal et entrer les commandes suivantes : (~est un raccourci vers le dossier principal de l'utilisateur)

```
cd ~
wget http://idav.ucdavis.edu/~okreylos/ResDev/Vrui/Build-Ubuntu.sh
bash Build-Ubuntu.sh
```

Le mot de passe sera demandé pour installer les librairies requises et compiler le logiciel Vrui VR Toolkit. A la fin de cette étape une fenêtre s'ouvrira avec un globe terrestre. Fermer cette fenêtre et retourner dans le terminal.

4) Installer le package du programme Kinect 3D Processing à l'aide des commandes suivantes :

```
cd ~/src
wget http://idav.ucdavis.edu/~okreylos/ResDev/Kinect/Kinect-3.2.tar.gz
tar xfz Kinect-3.2.tar.gz
cd Kinect-3.2
make
sudo make install
sudo make installudevrules
ls /usr/local/bin
```

Vérifier que dans la liste affichée dans le terminal après la dernière étape apparaissent les noms KinectUtil et RawKinectViewer

5) Installer le logiciel SARndbox à l'aide des commandes suivantes :

```
cd ~/src
wget http://idav.ucdavis.edu/~okreylos/ResDev/SARndbox/SARndbox-2.3.tar.gz
tar xfz SARndbox-2.3.tar.gz
cd SARndbox-2.3
make
ls ./bin
```

Vérifier que dans les dernières commandes de la liste affichée dans le terminal contiennent les noms CalibrateProjector et SARndbox.

6) Connecter la Kinect première génération à l'ordinateur et télécharger les paramètres internes de la Kinect dans le terminal à l'aide de la commande suivante :

```
sudo /usr/local/bin/KinectUtil getCalib 0
```

Entrer le mot de passe admin.

7) Aligner la Kinect de manière à ce que la Kinect capte toute la surface intérieure du bac. Utiliser l'application RawKinectViewer à l'aide des commandes suivantes :

```
cd ~/src/SARndbox-2.3
RawKinectViewer -compress 0
```

Lors de cette étape, une fenêtre s'ouvrira avec les 2 vidéos captées par la Kinect, à gauche la caméra infrarouge et à droite la caméra couleur. L'ajustement de la position de la Kinect doit se baser sur la vidéo infrarouge (à l'aide d'un objet à disposer dans les coins du bac) pas sur vidéo couleur comme leur champ est sensiblement différent.

8) Obtenir l'équation du plan de la Sandbox et l'écrire dans le fichier BoxLayout.txt en utilisant l'outil RawKinectViewer.

```
cd ~/src/SARndbox-2.3
pluma etc/SARndbox-2.3/BoxLayout.txt &
```

Dans l'outil RawKinectViewer, zoomer sur la vidéo infrarouge à l'aide du bouton déroulant de la souris, pour bouger dans la fenêtre presser la lettre z sur le clavier.
Dans cette fenêtre, maintenir le bouton dérouleur de la souris et déplacer le pointeur dans le menu qui s'est afficher sur « Extract Planes ». Ensuite, presser le bouton droit de la souris et déplacer le pointeur dans le menu qui s'affiche sur « Average Frame ». Placer le pointeur de la souris dans un coin la vidéo et maintenir le bouton dérouleur enfoncé et en même temps bouger le pointeur de la souris jusqu'à ce que le rectangle couvre l'entièreté de la surface, lâcher alors le bouton. Retourner dans le terminal, copier la dernière ligne correspondante à « Camera-Space plane équation » à la première ligne du fichier .txt en remplaçant le signe égal par une virgule.

9) Mesurer les coordonnées des 4 coins de la surface de sable à l'aide de l'outil RawKinectViewer.
Dans l'outil RawKinectViewer, zoomer sur le coin en bas à gauche de la vidéo infrarouge à l'aide du bouton déroulant de la souris, pour bouger dans la fenêtre presser la lettre z sur le clavier. Dans cette fenêtre, maintenir le bouton dérouleur de la souris et déplacer le pointeur dans le menu qui s'est afficher sur « 3D Positions ». Ensuite, presser le bouton droit de la souris et déplacer le pointeur dans le menu qui s'est afficher sur « Average Frame ». Placer le

pointeur de la souris dans un coin en bas à gauche de la vidéo sur le pixel coloré le plus éloignée et maintenir le bouton dérouleur enfoncer. Relâcher le bouton et refaire cette opération sur les 3 autres coins dans l'ordre suivant : coin en bas à droite, coin en haut à gauche et le coin en haut à droite. Retourner dans le terminal, copier les coordonnées des 4 coins dans le fichier BoxLayout.txt en dessous de l'équation de plan.

10) Connecter le vidéo projecteur à l'ordinateur via un port HDMI sur la carte mère, l'allumer et s'assurer que le champ couvre toute la surface du sable, ceci peut être facilité par l'utilisation de la commande suivante dans le terminal :

```
XBackground
```

Mettre la fenêtre en plein écran.

11) Calibrer le projecteur et la Kinect en utilisant l'application CalibrateProjector à l'aide des commande suivantes :

```
cd ~/src/SARndbox-2.3
./bin/CalibrateProjector -s <width> <height>
```

Où <width> et <height> sont les dimensions de l'image du projecteur en pixels.
Une fois la fenêtre ouverte, la mettre en plein-écran. Maintenir le bouton gauche de la souris appuyé, le pointeur dans le menu qui s'affiche sur « Capture Tool ». Appuyer alors sur le bouton de droite. Ensuite, à l'aide d'un disque en carton blanc de 15cm de diamètre avec 2 droites perpendiculaires passant par le centre, le placer le plus proche de la surface de manière à ce que la croix projetée coïncide à la croix sur le disque et que le disque apparaisse sur l'écran soit vert. Une fois vert appuyer sur le bouton de gauche. Et re-commencer l'opération en plaçant le disque en alternance un fois plus haut et une fois plus proche de la surface de sable. Au cours de cette procédure creuser une fois dans le sable pour capter le point le plus bas et appuyer sur le bouton de droite et continuer la procédure.
Si à la fin de cette étape, rien de se passe et qu'une erreur apparaît dans le terminal c'est que le disque n'était pas vert quand le bouton a été pressé, il est nécessaire de recommencer l'opération.

12) Ouvrir et exécuter la Sandbox avec les commandes suivantes :

```
cd ~/src/SARndbox-2.3
./bin/SARndbox -uhm -fpv
```

Si lors de cette étape, le rendu de la topographie ne couvre pas toute la surface du sable, alors recommencer les étapes 8 et 9.

Annexe 7 : Modification du code pour la représentation de lave

Ouvrir le fichier shader SurfaceAddWaterColor.fs dans le dossier du programme contenant les shaders avec comme chemin à partir du dossier principal Share<Sandbox-2.2<Shader .

Dé commenter la ligne 170 :

```
float  colorW=max(turb(vec3(fragCoord*0.05,waterAnimationTime*0.25)),0.0);  //  Turbulence noise
```

Commenter la ligne 177 :

```
// colorW=pow(dot(wn,normalize(vec3(0.075,0.075,1.0))),100.0)*1.0-0.0;
```

Dé-commenter la ligne 180 et commenter la ligne 179

```
//vec4 waterColor=vec4(colorW,colorW,1.0,1.0); // Water

vec4 waterColor=vec4(1.0-colorW,1.0-colorW*2.0,0.0,1.0); // Lava
```

Cette procédure peut être exécutée pendant que le programme est allumé.

Annexe 8 : Configuration de l'ordinateur utilisé avec le programme Wolf2D

Édition	Windows 10 Professionnel
Version	1607
Version du système d'exploitation	14393.1198
ID de produit	00331-20020-00000-AA352
Processeur	Intel(R) Core(TM) i7-3770K CPU @ 3.50GHz 3.50 GHz
Mémoire RAM installée	32.0 Go
Type du système	Système d'exploitation 64 bits, processeur x64

Annexe 9 : Complexité des librairies

Les tableaux 21 et 22 donnent les détails de la complexité pour les logiciels analysés dans cette étude à savoir SARndbox 2.3 et Kinect-3.2.

Ils sont suivis d'une explication des métriques utilisées.

File Name	Lines	Statements	Percent Branch Statements	Percent Lines with Comments	Classes Defined	Methods Implemented per Class	Average Statements per Method	Line Number of Most Complex Method*
Sandbox.cpp	1325	860	16,5	12	0	8,67	25,3	542
WaterTable2.cpp	956	633	5,1	18,4	0	11,5	25	388
SurfaceRenderer.cpp	921	369	13	17,8	0	9	18,1	93
HandExtractor.cpp	799	443	21,2	16,8	3	11	30	358
CalibrateProjector.cpp	754	458	18,6	12,5	2	3,8	21,7	687
BathymetrySaverTool.cpp	559	325	19,1	20,9	0	6,33	13,3	381
RainMaker.cpp	323	157	13,4	19,2	3	7,33	5,4	236
FrameFilter.cpp	312	173	19,7	20,2	0	12	13,1	34
DepthImageRenderer.cpp	302	176	10,8	21,9	0	6,5	11,5	96
WaterRenderer.cpp	188	120	5,8	17	0	2,5	19,8	112
DEMTool.cpp	169	72	9,7	17,8	0	7,5	3,3	91
LocalWaterTool.cpp	168	96	9,4	11,9	0	10	7,4	113
ElevationColorMap.cpp	151	82	15,9	15,9	0	7	9	59
DEM.cpp	137	72	5,6	16,1	0	5,5	4,6	62

Tableau 21 : Complexité du logiciel SARndbox 2.3

File Name	Lines	Statements	Percent Branch Statements	Percent Lines with Comments	Classes Defined	Methods Implemented per Class	Average Statements per Method	Line Number of Most Complex Method *
Camera.cpp	1344	798	18,2	21,8	1	10,5	16,5	1144
RawKinectViewer.cpp	1167	642	21,7	14	0	16	17,7	1018
KinectViewer.cpp	1015	530	24,3	16,7	0	5,25	10,6	749
GridTool.cpp	852	565	21,1	11,2	0	18	29,3	289
Projector2.cpp	816	350	14,9	15,4	0	11	14,1	103
CornerExtractor.cpp	788	471	22,3	19,7	2	7,67	18,7	310
DiskExtractor.cpp	747	431	15,5	19	6	8,67	14,2	301
KinectV2DepthStreamReader.cpp	720	415	24,3	20,6	1	12	32,7	168
Projector.cpp	691	382	24,1	18,1	0	10,5	16,8	270
CameraRealSense.cpp	620	365	17,5	14,7	0	24	13,2	519
AlignPoints2.cpp	601	368	18,5	12	4	3,67	22,6	491
KinectUtil.cpp	558	358	22,3	14,2	1	1	1	228
AlignPoints.cpp	471	310	23,2	8,3	1	2	58,5	329
DirectFrameSource.cpp	463	262	17,2	15,6	0	22	9,9	59
ShaderProjector.cpp	461	255	9,4	17,1	0	8	13,6	224
SphereExtractor.cpp	449	267	8,2	20	5	5,8	6,5	323
LineTool.cpp	404	249	24,9	10,4	0	13	17,2	172
TiePointTool.cpp	382	216	14,8	18,3	2	4,67	12,6	94
PlaneTool.cpp	379	212	25,5	10	0	7	26,9	81

SphereExtractorTool.cpp	373	224	13,4	14,5	0	9,5	9,8	327
KinectPlayer.cpp	365	188	17	17,5	0	5,33	9,2	309
FileFrameSource.cpp	363	184	26,1	14,3	0	18	8,4	167
MultiplexedFrameSource.cpp	338	167	20,4	17,8	0	6	12	198
NewCalibrateCameras.cpp	333	249	14,5	10,2	1	0	0	53
CameraV2.cpp	306	167	17,4	18,6	1	6,5	10,7	178
LWOWriter.cpp	284	164	9,1	16,2	0	0	0	43
KinectV2CommandDispatcher.cpp	254	132	8,3	20,5	0	14	7,6	42
KinectClient.cpp	250	124	17,7	14,4	0	6,5	6,7	50
KinectRecorder.cpp	240	118	14,4	22,1	0	3,67	7,6	234
KinectV2JpegStreamReader.cpp	240	128	14,8	21,3	0	11	9,8	125
DepthFrameWriter.cpp	235	73	15,1	15,7	0	5	11,8	202
DepthCorrectionTool.cpp	232	130	19,2	16,4	0	6	16,2	149
SpaceCarver.cpp	222	150	25,3	6,3	0	0	0	137
FrameSource.cpp	215	103	23,3	16,3	0	6	4,4	169
CalibrateCameras.cpp	207	116	21,6	11,6	0	0	0	34
FrameSaver.cpp	193	84	10,7	19,2	0	9	6,9	110
CalibrationCheckTool.cpp	187	57	7	11,2	0	8	5,1	174
MakeHuffmanTable.cpp	168	97	19,6	11,9	2	1	0	78
ColorFrameReader.cpp	164	77	13	15,9	0	3	20,3	92
DepthFrameReader.cpp	157	70	22,9	23,6	0	6	9,7	91
LossyDepthFrameReader.cpp	157	69	14,5	16,6	0	3	17,7	93
PointPlaneTool.cpp	157	64	18,8	12,7	0	6	8,2	73
CalibrateDepth.cpp	148	93	16,1	10,1	1	0	0	49

LensDistortion.cpp	142	72	16,7	21,8	0	13	4,2	52
LossyDepthFrameWriter.cpp	133	73	6,8	14,3	0	3	19	102
Renderer.cpp	115	40	20	13,9	0	9	2,8	37
ColorFrameWriter.cpp	113	55	5,5	15	0	3	12,7	46
DepthCompressionTest.cpp	107	68	20,6	10,3	0	0	0	32
LibRealSenseContext.cpp	106	43	14	16	1	5	5	42
MD5MeshAnimator.cpp	106	51	17,6	13,2	0	4	10	79
OpenDirectFrameSource.cpp	99	40	22,5	18,2	0	0	0	86
CompressDepthFile.cpp	97	61	19,7	10,3	0	0	0	34
MeasurementTool.cpp	96	36	13,9	14,6	0	6	3,7	94
HilbertCurve.cpp	87	33	24,2	13,8	0	4	6,8	32
KinectServer.cpp	85	31	9,7	20	0	2,5	3,6	52
TestAlignment.cpp	85	48	22,9	5,9	0	0	0	38
KinectServerMain.cpp	79	45	11,1	11,4	0	0	0	44
Motor.cpp	74	33	24,2	13,5	0	4	6,3	71
ColorCompressionTest.cpp	67	38	5,3	10,4	0	0	0	32
PauseTool.cpp	54	17	5,9	16,7	0	5	1,6	67
FrameWriter.cpp	32	6	0	9,4	0	2	1	32
FrameReader.cpp	27	3	0	11,1	0	1	0	32
KinectProtocol.cpp	24	3	0	12,5	0	0	0	

Tableau 22 : Complexité du logiciel Kinect-3.2

C++ Metrics

The metrics in C++ source files can be ambiguous when conditional compilation is used to define different versions of a substructure. Therefore, SourceMonitor ignores all conditional #else clauses and counts metrics only in the code that lies between the #if... and #else preprocessor directives.

Statements: in C++, computational statements are terminated with a semicolon character. Branches such as **if**, **for**, **while** and **goto** are also counted as statements. The exception control statements **try** and **catch** are also counted as statements. Preprocessor directives **#include**, **#define**, and **#undef** are counted as statements. All other preprocessor directives are ignored. In addition all statements between each **#else** or **#elif** statement and its closing **#endif** statement are ignored, to eliminate fractured block structures.

Percent Branch Statements: Statements that cause a break in the sequential execution of statements are counted separately. These are the following: **if**, **else**, **for**, **while**, **break**, **continue**, **goto**, **switch**, **case**, **default**, and **return**. Note that **do** is not counted because it is always followed by a **while**, which is counted. The try block statement **catch** is also counted as a branch statement.

Percent Lines with Comments: The lines that contain comments, either C style (/*...*/) or C++ style (//...) are counted and compared to the total number of lines in the file to compute this metric. If you select the project option to ignore headers and footers, contiguous comment lines at the beginning and end of files are **not** counted as comments.

Classes: Classes (including structs) are counted on the basis of their definitions. In general, classes are defined in header files while methods are usually implemented in separate files. Therefore the number of classes may be different from the number of classes for which a biggest method has been measured (see below). Template definitions are counted as class definitions.

Methods per Class: Both inline and non-inline class, struct, and template class implementations are counted. This metric is an overall average for all class, struct, and template method implementations in a file or checkpoint, computed as the total number of methods divided by the total number of classes and templates for which method implementations are found.

Functions: Number of functions found.

Average Statements per Method: The total number of statements found inside of methods found in a file or checkpoint divided by the number of methods found in the file or checkpoint.

Maximum Method or Function Complexity: The complexity value of the most complex method or function in a file. (Note: when the Modified Complexity option is enabled, average complexity values are displayed with a trailing asterisk. Example: "4.56*".) The Project and Checkpoint Views display only the method or method with maximum complexity; however, in the Details View for a file (double click on a file in Checkpoint View) the complexities of all methods and functions are listed, along with all method metrics. All method complexities are shown in the Method View as well.

Calls: The total number of calls to other methods or functions found inside of all methods or functions in a file or checkpoint are displayed in the Method View.

Maximum Block Depth: Blocks that define namespaces are not included in block depth metrics.