Kubb VR

TNM091 - Media Production for
Immersive Environments

Oskar Ankarberg
MSc in Media Technology and Engineering
Linköping University
Norrköping, Sweden
Email: oskan069@student.liu.se

I. Introduction

Research and inventions in Virtual Reality(VR) has recently paved way for a new generation of tools to develop VR experiences. The number of games available on Steam [1] (The largest game manager for HTC Vive experiences) as of today[1] is 1083 and new games are released almost every day. Steam is only one among many other platforms providing games for VR and more platforms starts supporting it. Oculus have 194 games available for their Oculus Rift and Playstation released VR support in October this year(2016).

With this recent increasing interest and technology development of VR an idea got introduced to create a virtual experience of playing the Swedish game Kubb. It is told that the game originates from Gotland [2] but no written references exists of the game before 1990s, when it turned into a popular summer game. The game consists of a field with 10 towers, 5 on each side, and 1 larger tower, called king, in the middle of the field. The game is then turned based with 6 throw sticks for each team on every turn. The designated winner is the first team to throw down the towers including the king of the opponent team.

Kubb relies upon a smooth and precise arm swing together with a release angle of the throw stick. This was the main reason for creating this game in HTC Vive since we would be able to regenerate the feeling of playing Kubb quite good in advance. The fact that the player does not need to make any complex movements relied on speed and precision, e.g golf, can be of great advantage for this VR-experience. HTC Vive provides a good and accurate system for tracking controls but a game with fast and complex movements will increase risks of developing a more error-prone software.

[1] 11/28/2016

II. Designing the VR-Experience

In order to design a good VR experience, visualization, sound, movement, interaction and logic are all of great importance. The first step was to draw an image of the scene in a work shop environment together with brain storming. This increases the chances of creating a scene interpreted closed to reality. Different sounds as well as interaction methods with the objects was discussed and decided. The goal of this VR-experience is to design a world perceived as a classic Swedish sunny summer evening at the country side. For those who have not experienced one, the environment usually consists of a forest, a sea, a beach, a village, boats and of course the sun dusk. These main characteristics together with things to spice up the country feeling e.g. a tractor, maypole and animals will then generate the visual result for the experience. The game engine Unity[3] was used to develop the experience.

A. The Visual

HTC Vive is a complex piece of technology composed into one headset and two controllers. The headset contains of two screens with 1080x1200 pixels per screen and a field of view of 110 degrees with a refresh rate of 90Hz. The disadvantage of these specs is a weight of 555 grams. This will reduce the feeling of reality already before starting the VR-experience and the idea of creating a lens flare to simulate wearing some sort of classes was tested. In unity flares can be defined on a light

source using an attached textured image. Although lens flares are created by different inner reflections in a lens, the feeling with lens flare was shown by user tests to be an efficient effect for this VR-experience.

B. The Sound

In [4] Serafin conducts user tests to see how images can be interpreted with sound by rating different places after hearing a sound. They show results that the human is good at estimating the environment according to sound. Hence, our sound environment must correspond to the visual scene to create a good VR-experience. The AudioSource in Unity comes with a predefined spatial 3D sound which uses the logarithmic function to determine sound level at a distance from sound source. This is used through out the project with some slightly modified distance functions to create a desired sound.

The following sounds are implemented as 3D spatial sound in the scene:

- Sound of tree clash. Generated from two pieces of wood knocked together.
- Sound from tree hitting the grass. Generated from a book landing on a couch.
- Sound from the flying seagulls. Downloaded from open source.
- Waves from the sea. Downloaded from open source.
- Sound from forest birds. Downloaded from open source.
- Sound from horses. Downloaded from open source.

C. The Logic

HTC Vive comes with a 4.5x4.5 meter bounding box that limits the user to move outside its bound. To move the bounding box a teleport functionality is implemented by casting a ray out from the controller. The ray determines where to move the bounding box by moving to the collided object. The player is further able to grab selected objects in the scene by using a collider around the controller and grabbable objects. The user grabs objects by tapping the trigger button located under the controller available for the forefinger.

The scene exploits Unitys physics engine to simulate gravity and collisions on objects. The physics characteristics of the kubb game are defined according to density of birch tree with the given volume of the objects. Material properties are custom selected to satisfy bouncing and sliding desired for the game. Although, this does not generate the randomness of small bumps in the grass as well as scratches on the tree objects but will suffice for now. The rest of the scene implements customized physics characteristics for desired functionality.

III. Methods

The game was developed in Unity 5 [3], a game engine which has support libraries for the HTC Vive. Fig. 1 shows the development of the environment, created using Unitys Terrain editor.

Fig. 1: Creating the environment in Unity 5

A. User Actions

Movements of the bounding box is as mentioned made by pointer actions, pushing the touch pad close to the thumb on the controller. The player further grabs an object by pushing the trigger button, see Fig. 6, Fig. 5. The tree stick is placed into the gripped hand, parallel direction to the controller while the controller turns invisible. A throw of an object is done by releasing the trigger button, the object then uses the controllers angular velocity to initialize the flying object. To reflect the feeling from throwing a real tree stick, the throw must correspond to the feeling, i.e weight, of throwing a controller. If the throw sticks (310 gram) calculated weight was used the user appeared to throw the stick too short in terms of predicted length from the user. The reason for this behaviour is the controllers weight of 200 gram and the physics of Unity simulating a throw of a 310 gram piece. To avoid this problem user tests pointed towards a multiplication of initial angular velocity by a factor of 1.2.

B. Bird Flocking

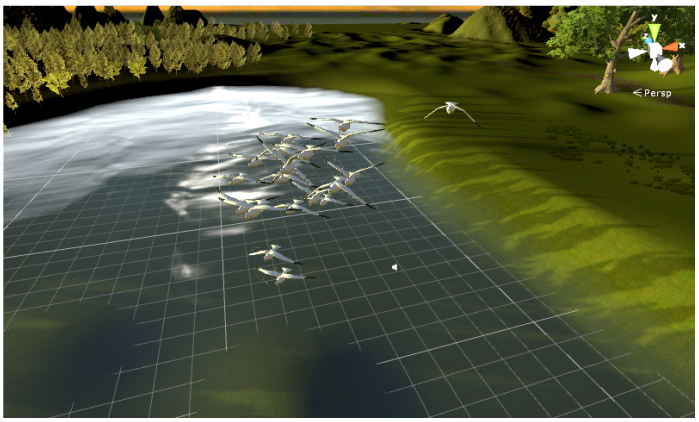

Fig. 2: Seagulls in Kubb VR

The seagulls in the game were implemented with Reynolds [5] algorithm. In [6] Hartman et. al describes the steps in terms of mathematical equations for easier implementation. The most important properties when implementing a flock behaviour are *cohesion*, *separation* and *alignment*. Let b_i represent a bird in the boid $B = \sum_{j=0}^{n} b_j$, then let b_j be the surrounding

birds where $b_j \neq b_i$ and $j = 0, 1, 2, .., n-1$ affecting b_i inside its visible radius V_i according to cohesion, separation and alignment. We further have the attributes: p_i position and v_i velocity.

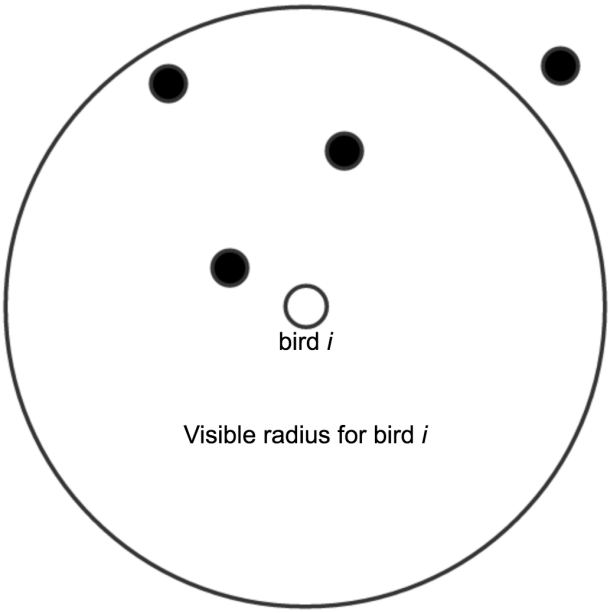

Fig. 3: Showing a bird and its visible radius V_i

- **Cohesion**
  Cohesion is the property contributing to one bird grouping into the surrounding ones with a force. c_i represents the visible birds average center position.

$$c_i = \sum_{b_j \in V_i} \frac{p_i}{m} \tag{1}$$

  The tendency of a boid b_i to navigate into the center of its visible radius V_i is then calculated as the cohesion displacement vector $\vec{k}$.

$$\vec{k} = c_i - p_i \tag{2}$$

- **Separation**
  Separation contributes to avoid collision between the birds. Its calculated as the negative sum of the visible birds position.

$$\vec{s} = -\sum_{b_j \in V_i} (p_i - p_j) \tag{3}$$

- **Alignment**
  The alignment property will force the birds to keep an uniform speed. It is calculated as the average velocity of the visible flock mates.

$$\vec{m} = \sum_{b_j \in V_j} \frac{\vec{v_j}}{m} \tag{4}$$

C. Water

The utilized water shader is *simple water* from Unity Standard Asset Package. Unfortunately it does not support stereo rendering for HTC Vive as of today, therefore reducing quality whilst increasing performance. The water sound point of origin is further implemented with an interpolation between points along the shore to determine the closest location to the player. This way we get a good estimation of how close the water is.

IV. Results

When the user starts the VR-Experience a menu is displayed within the world. Ambient sound from the sea and seagulls are present while the menu is displayed. The menu describes the VR-experience and how to exploit the functionality in the world. See Fig 4.

Fig. 4: Start menu of Kubb VR

Next, the user is free to play the kubb game by throwing the tree stick at the opponents towers. The user can collect all the sticks by pressing a button, to avoid walking around and throwing them back again.

V. Discussion

Creating a good VR-Experience proved to be a difficult task. Unity allows fast deployment of VR-experiences with a fast learning curve, but since its VR, it needs careful consideration when designing the world. The fact that two cameras needs to render the scene, one for each eye, decreases the performance. Another disadvantage of multiple cameras is the use of reflected materials implemented in Unity. The water reflections were rendered in mono and it was not possible to use Unitys predefined reflected water materials together with the HTC Vive. To render stereo reflections would be out of scope for this project. Hence, a simpler water shader was used without reflections.

VI. Future Work

In future versions of Kubb VR I would like to see a finished game play from start to end. Multiplayer option through network would also be an interesting feature to add or an additional static camera for the other player. The second player

would then use one of the controllers to throw from its side. More features can also be implemented to give it a more game like experience, e.g. data being displayed for the user about distance from aimed object, visual effects like fire around the tree sticks are also something to consider in the future.

VII. Conclusion

A VR-experience of the Swedish game Kubb has been created in the game engine Unity. Unity is an efficient pipeline to deploy VR-experiences but with some limitations regarding reflections for water with HTC Vive. Virtual reality is still limited to form a good experience of the real world and limited due to equipment. Weights and objects as well as smells can not be generated with the commercial technologies of today. Spatial sound together with correct visualization design is an important characteristics of VR to generate a feeling of presence. The result has fulfilled the goal of creating a first step towards a fully developed virtual reality of a Swedish summer evening with Kubb.

Appendix A
Images

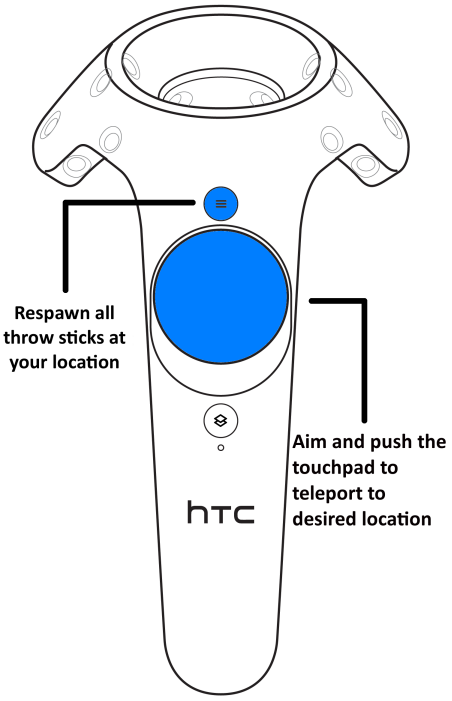

Fig. 5: HTC Vive controller, front buttons

Acknowledgment

I would like to thank Oskar Carlbaum for his work on modelling the Kubb king.

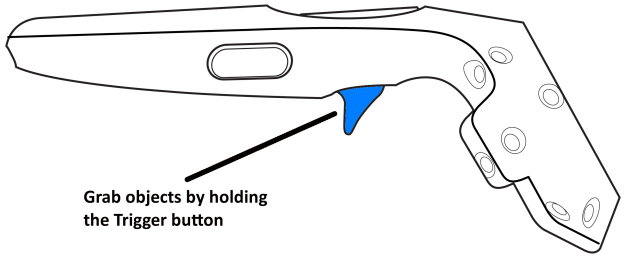

Fig. 6: HTC Vive controller, throw button

References

[1] Steam. URL: http://store.steampowered.com/search/?snr=1_4_4__12&term=vr (visited on 11/30/2016).

[2] Kubb. URL: https://sv.wikipedia.org/wiki/Kubb_(spel) (visited on 11/28/2016).

[3] Unity. URL: https://unity3d.com/ (visited on 12/14/2016).

[4] Stefania Serafin and Giovanni Serafin. *Sound Design To Enhance Presense in Photorealistic Virtual Reality*. 2004.

[5] Craig W. Reynolds. "Flocks, Herds and Schools: A Distributed Behavioral Model". In: *SIGGRAPH Comput. Graph.* 21.4 (Aug. 1987), pp. 25–34. ISSN: 0097-8930. DOI: 10.1145/37402.37406. URL: http://doi.acm.org/10.1145/37402.37406.

[6] Christopher Hartman and Bedrich Benes. "Autonomous boids". In: *Computer Animation and Virtual Worlds* 17.3-4 (2006), pp. 199–206. ISSN: 1546-427X. DOI: 10.1002/cav.123. URL: http://dx.doi.org/10.1002/cav.123.