EE267 Project Proposal: First-person Tower Defense Game for Mobile VR on Google Cardboard

Chi Zhang
czhang94@stanford.edu

Qian Yu
qiany@stanford.edu

October 9, 2017

Summary

In this proposal, we describe an idea and overall technical plan for developing a tower defense game for mobile virtual reality[1]. Our overall objective is to build an interactive, creative and entertaining first-person tower defense game, providing players with an highly immersive experience. Firstly, we briefly introduces background and motivation of this work. Then, some technical approaches of implementation are discussed. Lastly, we show a timeline with the tasks and milestones necessary to complete the project.

1 Background and Motivation

Tower defense(TD) is a subgenre of strategic video games where the goal is to defend a player's territories or possessions by obstructing the enemy attackers, usually achieved by placing defensive structures on or along their path of attack [3]. In traditional tower defense games, such as Plants vs. Zombies shown in Fig.1(a), strategic choices and positioning of defensive elements is an essential strategy. For the convenience of viewing the complete map and overall attacker deployment, most of them are designed as third-person perspective.

(a) Plants vs. Zombies (b) Beach Head 2002

Figure 1: Third-person and first-person tower defense game examples

However, there also exists first-person designs, such as Beach Head 2002 (see Fig.1(b)), in which the player controls the defenses in a bunker to hold off an enemy assault. This kind of games combine concepts of tower defense and experiences of first-person shooter(FPS) and takes advantage of great sense of participation and involvement induced by FPS. But the fact that player cannot change his position in the world leads to bad experience in the aspect of strategy.

[1]The mobile platform and VR device will be iOS running on Apple's iPhone and Google Cardboard.

Our idea is to develop a tower defense game where the player can change his bunker position and switch weapons to use in each bunker as well. In the sense of strategy, the player can choose enemies of which area to eliminate first by which kind of weapons. And as same as the majority of current VR games, our game design will also adopt first-person perspective since it creates better sense of immersion.

2 Proposed Technical Approaches

In this section, we will discuss two aspects of technical details based on currently popular practices in the domain of mobile VR development. Many of the approaches are used to overcome the computational limitation of mobile devices and achieve the best performance as well.

2.1 Unity game development

2.1.1 Baked lightmaps

One of the biggest issues regarding mobile devices is limited computational ability to render 3D scenes real-time. Real-time lighting requires nearly no preparations for good looking results but Unity needs to draw nearly twice the amount of geometry to calculate high quality lights and shadows. Although modern graphics hardware on newer mobile devices can handle real-time rendering better and better, they still suffer framerate drops in most cases. Thus, for mobile VR, developers need to avoid real-time lighting as much as possible.

Fortunately, Unity offers different options for precomputed lighting, one of which is baked lightmaps [2]. Lightmaps can be considered as textures that contain light information instead of color. Areas on an object which are in shadow are saved as darker pixels and lit areas are saved as lighter colored pixels. Those textures are mapped directly on objects during runtime, which makes lightmaps just as costly as an additional texture. Since it is precomputed, lightmaps affect the graphic performance much less than real-time lighting.

2.1.2 Navigation and Pathfinding

To fully implement a tower defense game, we propose to use Unity's intrinsic navigation system for enemies' attack pathfinding rather than fancy artificial intelligence techniques. The navigation system allows us to create characters that can intelligently move around the game world, using navigation meshes that are created automatically from our Scene geometry. Dynamic obstacles allow us to alter the navigation of the characters at runtime, while off-mesh links let us build specific actions like opening doors or jumping down from a ledge [1].

The module we are going to use is navigation mesh (NavMesh). The process of creating a NavMesh from the level geometry is called NavMesh Baking. The process collects the `Render Meshes` and `Terrains` of all `Game Objects` which are marked as *Navigation Static*, and then processes them to create a navigation mesh that approximates the walkable surfaces of the level. By setting all of viewpoints to be destinations of navigation, we can obtain fairly great performance on attackers' automatic pathfinding.

2.2 Immersive virtual reality via Google Cardboard

2.2.1 Google VR SDK

Google VR SDK implements most of VR components in unity, such as stereo distortion, lens distortion, etc. This makes VR development a lot easier and we only need to focus on the scenes and scripts in Unity. Furthermore, Google VR SDK also supports mobile VR. We can deploy our VR game to iOS and Android devices conveniently.

2.2.2 Mobile VR hardware

Professional VR devices like HTC Hive and Oculus Rift are very expensive nowadays. In contrast, mobile VR is more realistic since everyone affords a smart phone. What's more, most of mobile phone integrates IMU, accelerator and magnetometer, and Google VR SDK handles orientation tracking very well. However, the challenge of mobile VR is its computation power. We have to sacrifice a lot of rendering quality on mobile VR as well as part of game logic.

3 Timeline

Please refer to the table below for detailed milestones and plans of this project.

Checkpoints	Goals
5/27	3D scene construction
5/30	Vanilla version of one-point tower defense
6/2	Add multiple viewpoints to the scene
6/4	Implement weapon-switching functionality
6/5	Add prefabs, models and animation
6/6	Bake lightmap and add navigation component
6/7	Work on demonstration examples
6/8	Finalize project deliverables

Table 1: Tentative schedule for the project

References

[1] Unity Manual. Navigation and pathfinding, 2017. [Online; accessed 25-May-2017].

[2] Unity Manual. Using precomputed lighting, 2017. [Online; accessed 25-May-2017].

[3] Wikipedia. Tower defense — wikipedia, the free encyclopedia, 2017. [Online; accessed 25-May-2017].