

Hostile Takeover: A critique

By Mathew Hughes



For the second Digital Media Environments project, I chose to develop a short 3d game. The user is able to interact with the game environment through the use of a Nintendo Wii remote.

The game is set in a fictional office space, that for the purpose of the game, has been overrun with cardboard cut-out targets. The aim of the game is to travel through the office and shoot the targets, with each target hit in this way increasing the players score. This is accomplished by pointing the Wii remote at the screen and pressing the trigger button, which fires a tennis ball into the environment. In this way the Wii remote simulates a gun (I would also be using the Wii Zapper to add to this effect), and merges the physical environment with the digital one. It's worth noting I did not want to use the motion sensing capabilities of the Wii remote, just the pointer functionality. I chose to shoot tennis balls, opposed to bullets, as the physics of the tennis ball bouncing around and interacting with office furniture seemed like an interesting concept. It also helps to keep the game light-hearted.



The Wii remote and nunchuk in the Zapper housing

I chose the setting of the game to be an office, as they are generally small and enclosed environments, making a good contrast to the wide open environment of my previous "Digital Castle" project. They also offer a large selection of furniture and other items to be modelled and placed within the game.

The tools I chose to develop the project with were 3ds Max and Unity 3D. I chose 3ds Max, as it was a 3d modelling tool I had previously used in the earlier project and so had experience with. I chose Unity 3D as a game engine, as there is a free version available and also a large and active community of developers to offer support for any problems I would encounter during development.

Early on in development I decided that I wanted the game to be based in a first person perspective. I chose this as it made the most sense if the player would be pointing the Wii remote at the screen and also due to First Person Shooters (FPS) being a common genre. I was also debating whether to

allow the player to walk around the game environment (using the Wii remote Nunchuk attachment) or to make the players movement automatic (or "on-rails"). I chose to make the game on-rails for reasons I will explain later in the critique.

My first task was to get the Wii remote interacting with Unity. I found a post in the Unity forums about a plug-in that enables the use of the Wii remote. After downloading the plug-in I realised that use of plug-ins was disabled within the free version of Unity and that Unity Pro was needed. However there was a free trial of Unity Pro available for 30 days. Once the trial and plug-in were installed, I attempted to connect to Wii remote to the pc.

This was harder than I first thought. The Wii remote plug-in for Unity was originally written for Mac OS X. As I use Windows, I was having to use the Windows version of the plug-in. However much of the documentation still referred to the Mac version of the plug-in. Also the plug-in attempted to use a built in Bluetooth device and not my USB dongle. On top of this the default Windows Bluetooth manager was unable to recognise the Wii remote. Eventually I found that I had to connect the Wii remote via a third party Bluetooth stack before attempting to run the Unity app. Unfortunately this solution appeared to only work intermittently and gave very mixed results. Sometimes the test app bundled with the plug-in seemed to work as intended. Others it partially worked and others it failed to work at all.

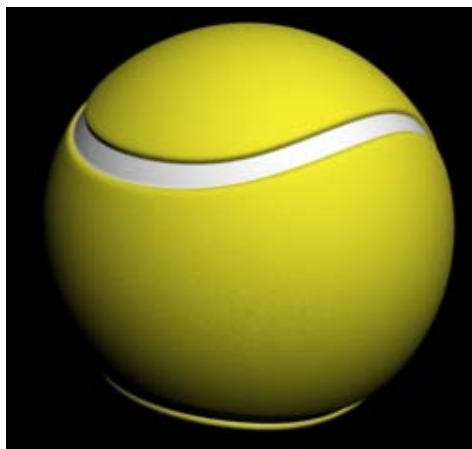
With the lack of documentation for the windows platform and the mixed results, I decided to abandon using the plug-in and find another way to connect the remote to the pc.

After searching various websites, I found a program called 'WiinRemote'. This program interpreted input from the Wii remote and translated it in a way that allowed the computer to understand. Using this program, I was able to control the on screen cursor using the pointer of the Wii remote and map the arrow keys on the remote to the arrow keys on the key board. In this way I hoped to easily be able to use the FPS controller built in to Unity. At this point I used the in built test game in Unity to try the functionality of the remote.

Again problems arose during the testing. Although the Wii remote's arrow keys could indeed move the in game character and the pointer control the cursor on the screen, the character could not look around the world. I had already spent a lot of time trying to get the remote functioning in a way that would allow me to create a more traditional FPS game, I decided to make the players movement scripted and give the game the on-rails feel I mentioned earlier.

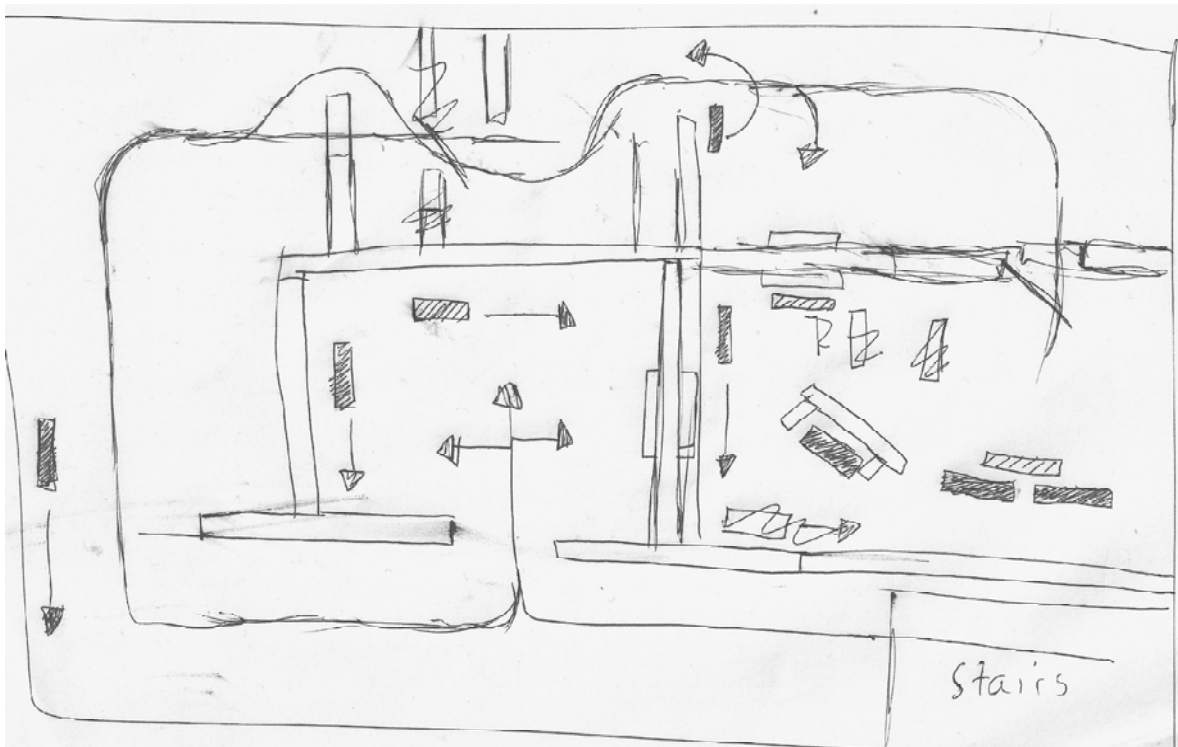
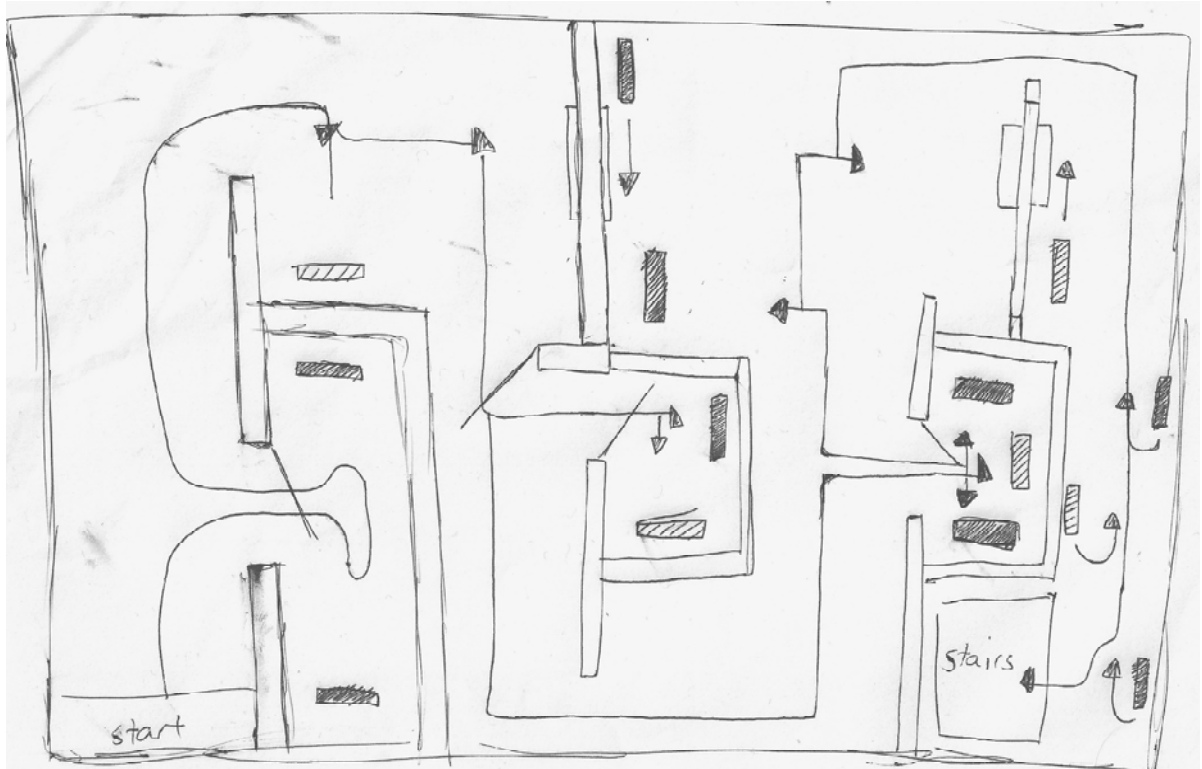
My next step was to find a method of animating the camera. Again by looking at the Unity forums I was able to find a couple of scripts to achieve this. One of the scripts attaches to the camera object, whilst the other attaches to a series of waypoints. Each waypoint had to be an empty game object and placed within the game world. It was then possible to use the inspector panel of the camera object to define the order I wanted the camera to travel to and how long it should take to get there. This worked well at first, but later on in the development of the project I found that I had far too many waypoints to manage effectively. The solution to this problem was to animate a small cone in the 3ds max file of the office. Using the cone as a mock camera I animated the positions I wanted the camera to be in. Once the 3ds file was imported into unity, I could then attach the camera object to the cone. By doing this the camera followed the animation path of the cone.

Next I wanted to get the shooting aspect of the game working. To do this I adapted the FPS tutorial available from the Unity website. The tutorial itself described how to make a gun fire objects directly in the middle of the screen. Using this as a basis I was able to adapt it so that the gun pointed towards the mouse and fired the object in that direction. By following a tutorial from YouTube, I was able to sculpt a tennis ball and use this as my object to shoot. Adding physics to the ball within Unity was easy and even included a 'bouncy' physics preset, meaning the tennis ball bounced from objects realistically.



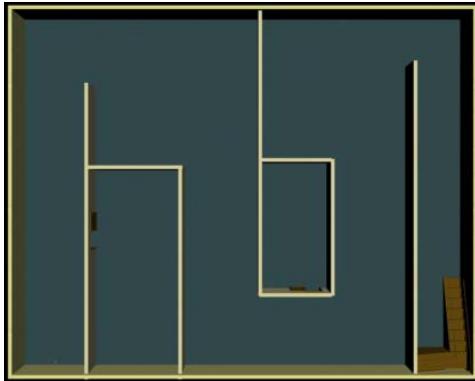
A 3d render of my tennis ball

With a basic camera set up and the Wii remote working as I needed, I moved onto modelling the environment I wanted to use. Below you can see my initial sketches of the floor plan of my office.

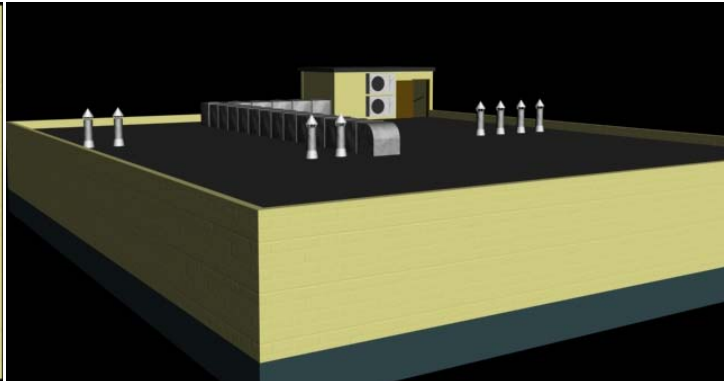


Initial sketches of the office floor plan. Black boxes represent targets that if hit score points. Dashed boxes represent targets that if hit reduce points (an element I later dropped). Arrows represent the movement of the player and targets (moving targets was another feature I dropped).

However during the modelling of the building my ideas changed somewhat. The ground floor design changed slightly and the second floor became the roof of the building. I decided to make this change as my original designs were too complicated.



3d render of the office ground floor
as seen from above



3d render of the office roof

With a basic environment created I added a few targets to the scene. These targets were simply tall and thin rectangles similar to a target you might find at a shooting range. With a few targets in position I was able to test how the game played. I needed to make a few tweaks to the physics settings of the objects, such as decreasing the mass of the targets and increasing the mass of the tennis balls. Although this made the mass of the items not very realistic, the game played better, as the balls sent the targets flying around the room with an accurate shot.



Image of my target

With the basics of the game working and an empty environment completed, I set about filling the office with objects. The first object I created was a desk. The desk was simply a few thin rectangles and a sliced tube for a handle.



Render of the desk without any textures

Render of the desk with textures

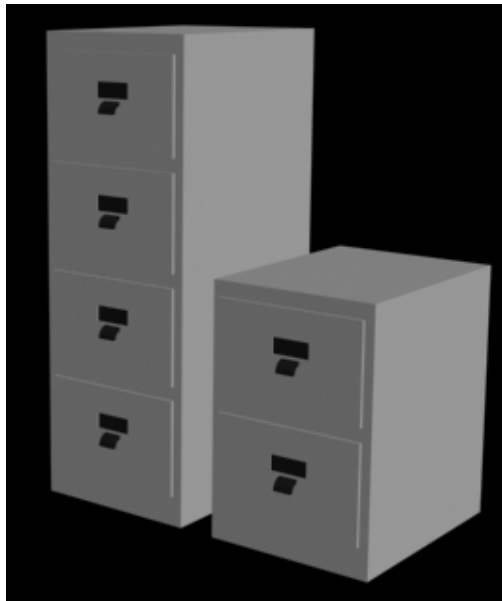
With the desk complete I moved on to creating an office chair. This proved more of a challenge and creating the arms proved one of the harder things to model in the project. I settled on creating the arms from three separate rectangles which I then shaped by hand and later merged them together.



Render of the chair without any textures

Render of the chair with textures

Next I moved on to creating a filing cabinet. I decided to have two sizes, a small one, with only two draws and a large one, with four draws. In contrast to creating the office chair, this was the easiest piece of furniture to produce, with the model consisting of only a few boxes and sliced tubes to create handles, in the same way I had done for the desk.

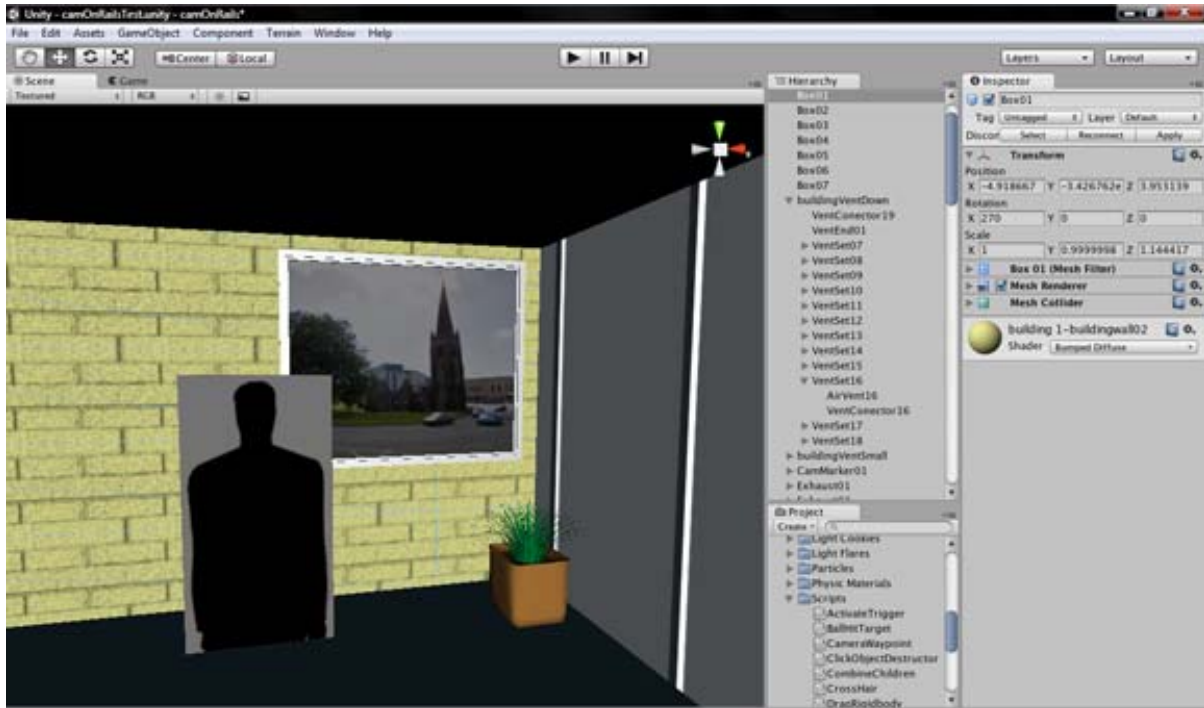


The completed filing cabinets

Other objects I created included a photocopier, lights, notice boards, a small desk plant, both a food and a drinks vending machine and a water cooler. The water cooler proved to be the hardest thing to render effectively, as I wanted to give the impression that the bottle was full with water, but also had to be simple enough to be included in the game engine.



To give the outside wall of the office a more interesting appearance I created some windows. I then used pictures from various areas of Plymouth to become the texture for the glass of the window. In this way, it almost looks like that the office is located in Plymouth, and gives the game a local feel.



Screenshot of the Unity editor showing the window, a target and a plant

With everything I wanted to appear in the game created, I began to focus on putting it all together. Importing the objects into Unity was easy, but did create a few problems with the scale of the objects. This was easily fixed by adjusting the scale property of the files once they were imported. With the objects placed in to the environment, I noticed many of the textures I had set up in 3ds Max were lost during the importing process. Although the colours I had assigned to the objects were correct, effects such as bump maps was self illumination were lost.

After again searching the Unity forums for an answer, I found that the way 3ds Max assigns many of its settings is incompatible with Unity. The solution was to reapply the lost effects in Unity. Although this was a simple process, it was annoying that I had to do it again. It did however give me a better insight in to how different 3d programs handle texturing differently.

With the majority of the project now completed, I started to implement a basic scoring system. This was harder than I first thought, mostly because I was unfamiliar with scripting in Unity. However by using the community and consulting the documentation, I was able to create a system that

increased the player's score by 10 for each frame that a ball is in contact with a target. With this task finished, the project was complete.

Over all I am happy with the way the project has turned out. Some aspects were very challenging to complete to a good standard, but I have learnt a lot in the process. Parts of the game work better than others, for example the tennis balls sometimes go through objects instead of hitting and bouncing off of them. This problem is easily fixed by adjusting the interval that the collider events are checked for. However, by doing this the performance of the game suffered, so I had to find a balance that checked often enough that the physics were acceptable and that performance was not affected too much. Of course this is only a problem due to having to use my laptop for the presentation. If I was able to use a more powerful computer, the interval could be lowered with no loss of performance. If I were to do the project again, I would do more testing of the Unity engine to find out more about its capabilities and the way it works before jumping in and starting the project.



A screen shot of the game in progress.

References

Below is a list of websites, tutorials and guides I used during this project.

Tutorials and guides.

Various resources found on the Unity website (www.unity3d.com), but in particular:

- <http://download.unity3d.com/support/Tutorials/2%20-%20Scripting%20Tutorial.pdf>
- <http://unity3d.com/support/resources/tutorials/fpstutorial>
- <http://answers.unity3d.com/>
- <http://forum.unity3d.com/>

<http://www.youtube.com/watch?v=fo15pB2VtqU>

<http://3dstudiomaxtutorials.com>

<http://tutorialized.com>

<http://3dtotal.com>

Reference Material

<http://www.youtube.com/watch?v=ZSJj281AKk>

<http://www.youtube.com/watch?v=JaVpZsooc9A&NR=1>