**Prototype report**

Thomas Broens

Player movement and interface scripts:

At the moment we have implemented basic player movement we want to use in our game. These movements consist of walking, jumping and running. We implemented these scripts on a sphere object. Getting this to work didn’t take a lot of time because of previous experience with unity in the smaller project we did before. It is working just as we want it to, so we will most likely use it in the final game. The next step was to create an interface where we could see our player health and stamina. We did this by creating a health bar and stamina bar on the bottom left of the screen. Stamina decreases every time a player sprints or jumps. When sprinting the bar decreases over the time the sprint button is pressed down, while with jumping a set amount of stamina is lost when the jump button is pressed. Even though this now works to our satisfaction, it took a bit more time than expected. The stamina bar also regenerates slowly over time. As it is working fine, we decided that we will use it in our final game. Further improvement to these mechanics will be in the form of making them work correctly with our 3D models and animations.

Fighting script:

Another prototype is the melee combat system. At the moment it works as we want it to, but we aren’t sure yet if it is how we want our final combat system to work. We will keep using it for the coming time, but in my opinion we will need to make some improvement to make it viable for our end product. In what ways exactly I’m not sure yet, but I’m sure it’s shortcomings will come to light in the near future. To make these scripts work we also had to create an enemy with health. At the moment two spheres that collide (get in range) with the rights tags (being “Player” and “Nerd”) and the attack button is pushed the Nerd takes damage. The same goes for the player, except the enemy automatically attacks when they are in range. This is represented with the health bar of the player decreasing by a pre-set amount of damage. At the moment almost all the stats, as sprinting, damage given and taken, and the amount of stamina actions cost, etc. are publicly defined and can be changed. We are still looking for the best combinations. The enemy also has a movement script that uses a nev mesh agent which makes the enemy follow the players position. We will definitely not be using this mechanic in our end game, but until our AI system isn’t working properly it will serve its purpose just fine. Making it able to test different things as the walking and attacking animations of our enemy’s, and different scripts as fighting and taking damage.

These scripts where made and “optimized” over the last two weeks. Each day getting a little further but also having to adjust previously made scripts to make it al work. Al in al it did take a little more time than I expected, but as I’m working more and more with Unity, I notice that things are going smoother and faster. I think we have a solid base to continue with and am happy with the current prototypes.

Ajdin Husic

ANN level 1

At the moment, our ANN prototype is only a concept, and can be implemented as soon as the project possesses a CPU-controlled enemy player and a main player, manually controlled with all of its movements and actions available.

The ANN-prototype is described as a network containing at least six input neurons, six hidden neurons in one hidden layer and finally five output neurons in the output layer.

The states of the character will be translated to inputs/outputs using numerical values depending on the actions performed by the CPU, and the events handled by the user. Since the inputs and outputs are defined by numerical values, data will be sampled by letting the user control the enemy, and the original player’s character will be controlled by the CPU. The sampled data will be used in MATLAB and the weights will be trained accordingly.

Finally, the trained weights will be used to perform the enemy actions from now on, and at least one level of intelligence will be available.

Since the ANN is not yet to be implemented, the current concept’s testing period will be extended.

When will an Enemy attack?

The distance between an arbitrary enemy and the player will be computed as a floating point value. As soon as this value becomes smaller than a predefined constant floating point, the corresponding enemy will be allowed to attack (logical selection structure). As this Boolean obeys the logical condition, it will never revert to its original value, since this would allow the enemy to not care anymore as soon as the player gets too far away. This means that the enemy will then stay in “attacking mode”, after the player would come too near.

Also, regardless of the distance, the enemy will attack if it’s being attacked by the player first (e.g. by using a firing weapon). Note that the enemy will of course choose to perform certain actions in correspondence with the trained neural network.

3D Building Model

A building is created as a 3D model, which could be used for the world design. This model could be used to procedurally generate the levels. It will be discussed whether we will use procedurally generated models at all, or just create non-editable 3D models as a prefab, and only generate the world instead. The first created model is shown below.

