Linnéuniversitetet
Kalmar Växjö

1DV437 - Assignment 3

Technical report of my game

Author: Eric Enoksson
Semester: Summer -19
Area: Computer Science
Course code: 1DV437

Table of contents

Architecture **3**

Design patterns and data structures **3**

How collisions and geometry was handled **3**

Textures, shaders, materials and lighting **4**

Animations **4**

Architecture

I started with a modular architecture since I thought that was the simplest approach, being unfamiliar with unity, and not having the full picture of the project in my mind. I tried to avoid cyclic dependencies where I could, but this could for sure be improved at some places. I did not bother with message sending or observers or similar. At the root I have a game controller which is more or less the all-knowing class, controlling the big parts of the game.

Design patterns and data structures

Other than lists and arrays, Dictionaries were used for screen messages and level names, tying e.g. a scene build index to a level name. I made a Object/Bullet Instantiator (which I ended up not using), to preload and re-use bullets and impact effects that I implemented using a Stack. The stack would contain "free-to-use" objects that were popped when used and pushed back when not needed.

I used singletons in a couple of classes when I only needed one object of the class, making accessing easy and potentially reducing unnecessary coupling.

I extensively used instance variables instead of local variables to avoid creating excessive amounts of garbage and reducing memory allocation, especially when variables where read or written to every frame.

Polymorphism allowed me to keep the code cleaner, share common attributes and separate unique attributes. For example I used base classes for both enemies and the player where I put common properties, and then I made classes that inherited from the base class where they started to differ.

How collisions and geometry was handled

I used rigidbodies with suitable colliders for moving objects and object affected by physics. For the combatants, I locked x and z-rotations (since it's a top down game), and also movement in the y-axis to avoid potentially unwanted behaviour since they only move in two dimensions. Instead of using mesh colliders, I ended up using a simpler capsule collider for the combatants, since I couldn't get the mesh colliders to work properly.

To avoid bumping off the enemies from patrol paths and getting them stuck, I made them kinematic (and made sure the patrol path did not take them through a wall or similar.

For the player, I had to use physics overlapsphere, since I had problems glitching through walls and others objects. I made the overlapsphere slightly smaller than the collider and made sure to check collisions each fixed update before any movement was done. If the overlapsphere contains a collider, the movement is denied.

For bullet impacts I had to use raycasting. Since I wanted the bullets to have an adjustable travel time, I had to raycast every time it moved from the previous position of the bullet to the new position and check for collisions. If the bullet hit a rigidbody object, I moved the bullet to the point of impact to make sure the colliding object was affected by the force of the

bullet.

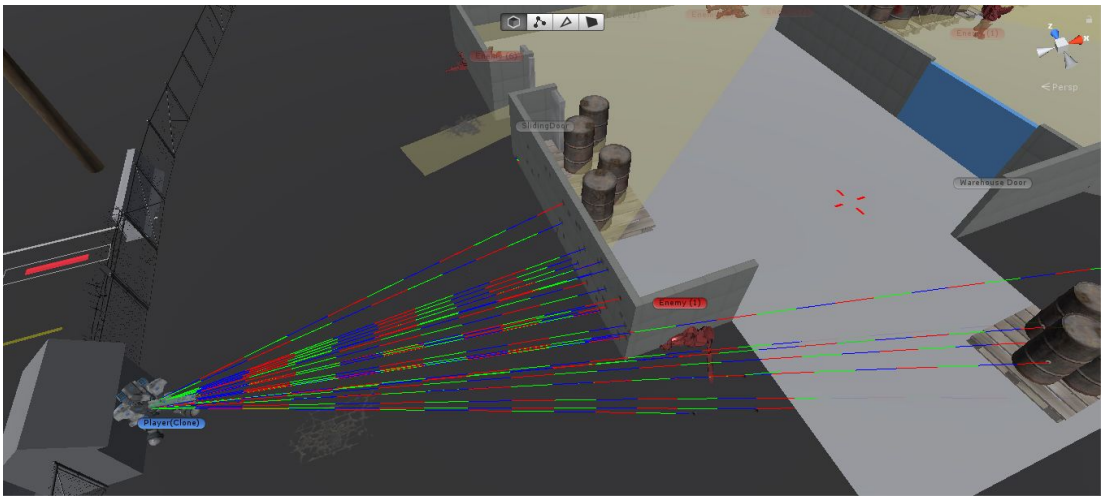

Figure 2. Debug lines used to visually see the effect of raycasting when firing. Each coloured line represents the distance a bullet travels during a fixed update.

Field of view mesh/target detection, and melee attacks also uses overlapsphere + raycasting with masks.

Textures, shaders, materials and lighting

I used a mix of basic coloured materials I made myself and those that came with the assets with texture images. To some materials I added emission in order to make it more visible, like e.g. a door release button. In all scenes I used a directional and a fill light to represent the daylight with light reflections. The rest of the lighting consisted of realtime spotlights and point lights.

Mainly, standard shaders were used, or whatever came with the assets.

I used probuilder to create geometry like floors and walls, which allows you to modify the size of objects rather than their scale, so textures won't stretch.

Animations

I used the animator for the combatant model. For the player, I created a new animator to get smoother movements and made a blend tree for the movement animations. It's manipulated by changing a float value based on player inputs. Some animations were played in isolation.

For enemy turning and door movements, I used Vector and Quaternion linear interpolation. Mostly I used these in conjunction with coroutines.