**Text, logo

Description automatically generatedIN3026 Advanced Games Technology**

**Coursework Report**

Archie Short / [archie.short@city.ac.uk](mailto:archie.short@city.ac.uk)

Table of Contents

[Introduction and Overview 2](#_Toc90144309)

[Assets and libraries 3](#_Toc90144310)

[Self Created Assets 3](#_Toc90144311)

[Assets from online 3](#_Toc90144312)

[Libraries Used 4](#_Toc90144313)

[Part 1 – Basic Game Modelling 5](#_Toc90144314)

[Intro Screen 5](#_Toc90144315)

[Primitive Objects 5](#_Toc90144316)

[Skybox and Terrain 6](#_Toc90144317)

[Audio 6](#_Toc90144318)

[HUD 6](#_Toc90144319)

[Part 2 – Camera, Meshes, Lighting, FX 7](#_Toc90144320)

[Camera Motion 7](#_Toc90144321)

[Mesh Objects 7](#_Toc90144322)

[Lighting and colouring effects 8](#_Toc90144323)

[Special Effects 8](#_Toc90144324)

[Part 3 – Physics, AI, Gameplay 9](#_Toc90144325)

[Game physics 9](#_Toc90144326)

[AI 10](#_Toc90144327)

[Gameplay Elements 12](#_Toc90144328)

[Variable Difficulty 12](#_Toc90144329)

[Reflection 13](#_Toc90144330)

[References 14](#_Toc90144331)

# Introduction and Overview

The name for my project is covert crab. You take the role of a crab super soldier, being trained to fight in the great crab wars. The game takes place inside a combat training simulation where your objective is to keep the beacons running while waves of enemies try to kill you and keep them disabled. The theme of the game is to be that of a simulation combat exercise, the music skybox and lighting all attempts to reflect that. In terms of genre my game is a first-person movement shooter. Similar to games such as dusk and quake where you have to stay constantly moving and shooting to stay alive. There are also elements that could fit the description “bullet hell” as the projectiles are at a speed you can dodge but that was not my initial intention.

Gameplay can be broken down into two elements, objective and survival. Around the map there are three beacons, each beacon has several switches that need to be activated and stay active to progress which is the objective part as engineer type enemies will go around turning them off again. As you complete these objectives there are enemies who do not care to disable the switches, they only have one goal in mind, killing the player. This is where the survival part comes into play as you must shoot the enemies while dodging their own attacks.

There are 5 enemy types in the game that are there to hinder you, the most important is the engineer who will disable the switches but not do any direct harm. Instead, they place down turrets which are stationary however if you enter their range they release a barrage of projectiles in your direction. The rest of the enemies all have the single objective of killing you. If you enter range they will pursue you until you leave it again. The enemies with these behaviours are the shotgun, flintlock and melee crabs. There is only one way to win the level but two ways to lose. To win all the beacons must reach 100% however if you lose all your health or the timer runs out then its game over.

In terms of tools at the players disposal there are 4; firstly, your movement which can be accelerated by sprinting however you can only sprint left and right as you are also a crab. Your flintlock pistol which can fire two shots a second, your cameras which you can place around the map to quickly check an area and see what switches are on or off and finally your interaction. Pressing E will create an interaction hitbox to interact with the switches around the map turning them on or off.

# 

# 

# Assets and libraries

All assets created with **Magica Voxel** are made with a free tool created by developer Ephtracy.

<https://ephtracy.github.io/>

## Self Created Assets

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Asset | Software made in | How it was created |
| Mesh | Flintlock | Magica voxel | Using references of a flintlock from google |
| Mesh | Bullet | Magica voxel | Making a squarish pellet |
| Mesh | Beacon | Magica voxel | Just a rectangle with details |
| Mesh | Enemy Engineer | Magica voxel | Prototyped a model for a crab back and fourth over the course of the module then added props |
| Mesh | Enemy Brute | Magica voxel | ^^ |
| Mesh | Enemy Shotgun | Magica voxel | ^^ |
| Mesh | Enemy Flintlock | Magica voxel | ^^ |
| Mesh | Letters | Magica voxel | Using a font I created previously for 3d text in magicavoxel, changing the colour scheme |
| Textures | Progress bar colours | MS Paint | Just changing the RGB colours |

## Assets from online

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Type | Asset | Date Accessed | Link | Author | Licence |
| Music | Level\_music2.mp3 | 8/12/2021 | https://www.youtube.com/channel/UC\_6hQy4elsyHhCOskZo0U5g | White bat audio | Royalty free no copyright |
| Music | Menu\_music.mp3 | 8/12/2021 | https://www.youtube.com/channel/UC\_6hQy4elsyHhCOskZo0U5g | White bat audio | Royalty free no copyright |
| Sound | Musket\_shot.wav | 8/12/2021 | https://audiojungle.net/item/flintlock-pistol-single-shot/12722263 | Sound-ideas | SFX (Single Use) License |
| Sound | Pick\_up.wav | 8/12/2021 | https://freesound.org/people/Leszek\_Szary/sounds/146727/ | [Leszek\_Szary](https://freesound.org/people/Leszek_Szary/) | Creative Commons 0 licence |
| Sound | Turret\_click.mp3 | 8/12/2021 | https://mixkit.co/free-sound-effects/click/ | n/a | Mixkit Licence |
| Font | Game\_font.ttf | 5/12/2021 | https://www.dafont.com/andrew-tyler.d2526?l%5b%5d=10&l%5b%5d=1&l%5b%5d=6 | Andrew Tyler | Free for personal use |
| Texture | Circuit.bmp | 5/11/2021 | [http://users.csc.calpoly .edu/%7Ezwood/teaching/csc471 /finalW1](http://users.csc.calpoly.edu/%7Ezwood/teaching/csc471/finalW1)  4\_1/fdhansen/ | Forrest Hansen | Open source |
| Texture | Terrain.bmp | 7/12/2021 | https://opengameart.org/content/sand-desert-dune-tile | kbmsfx | Free license |
| Texture | Skybox | 7/12/2021 | https://opengameart.org/content/elyvisions-skyboxes?page=1 | elyvisions | Free license |
| Textures | Explosion.tga | 9/12/2021 | https://opengameart.org/content/pixel-explosion-12-frames | JROB774 | Free license |

## Libraries Used

* Bullet
* assimp
* FMOD
* Freetype
* Glad
* GLFW
* Glm
* Spdlog
* Stb\_image

# Part 1 – Basic Game Modelling

## Intro Screen

My main menu is made of quite a few elements that come together to introduce the game. The controls are shown via a bitmap image rendered on the top layer. The lettering for the menu itself is made of individual meshes, each one is an imported object with their own physics and hitboxes.

This is because after a few seconds I have about 40 crabs come rushing onto the screen to mess up the title and steal the letters. These NPCs are a different class to the normal enemies which will be discussed later and just use a very basic state machine. They pick a point in the middle of the screen and move to it once there they choose a new point approach and repeat. The crabs have a much larger mass then the letters, so it gives a fun physics effect where the letters get tossed around.

A picture containing graphical user interface

Description automatically generatedA picture containing arrow

Description automatically generated

## Primitive Objects

The three primitive objects I created for my game take the roles of a turret, switch and power up. All have appropriate scaling, normal mapping and texture coordinates however I have only given one of the objects a texture in the final game as a style choice. On both the switch and turret primitive objects they have different states in which I use colour to give the player information and having a texture makes it a lot less clear in terms of what is going on. On the powerup however there is a texture and I’ve provided screenshots of what textures on the other two look like just to show that they can be applied correctly I’ve just chosen to leave it out of the game.

A picture containing text, outdoor, outdoor object

Description automatically generatedA picture containing light

Description automatically generatedA lamp on a table

Description automatically generated with low confidenceA picture containing floor, indoor, curtain

Description automatically generatedA picture containing floor, indoor, colorful

Description automatically generated

**A picture containing tent, outdoor object

Description automatically generatedA picture containing text

Description automatically generated**

Each of the primitive objects show appropriate rotation scale and transformation, all get placed dynamically during the runtime of the game and the have correct rotations. They are scaled to fit the scale of the rest of the game and not seem out of place. The turret rotates as well to wherever the player is when they are in range.

## Skybox and Terrain

The terrain consists of three game objects, one is for the menu and has a plain blue colour texture and the other two are for the level. These have a wrapped sand texture and one is the heightmap to lay out the bounds of the level and the other used for the physics as a ground. I used a crater heightmap then edited the code to smoothen the y to a flat surface beyond a certain depth to make it flat.

For the skybox I just changed it to an appropriate texture for the theme of the game whilst also making it bigger, so it didn’t cut off the edge of my level. As I increased the size I decided it would be best to reduce the y position a little too just to get the right look for the game.



## Audio

I chose the music in my game to fit the theme of the simulation aesthetic. There is a different track for the menu and the level and they change when you press space at the start. The track for the level is the same amount of time as the countdown in the top right so when you run out of time the song naturally ends too.

For my sounds synched to game events I did one for my players gunshot, collecting a pickup and when an enemy places a turret. They are very simple implementations to meet the requirement however I would say the turret placement noise does have a positive effect on gameplay. It lets you know there is an engineer enemy active and roaming around somewhere which is important to know as you want to stop them quick before they turn off all the switches you’ve turned on so getting an idea of their location from the turret placement click works well.

## HUD

My HUD is split between a few elements. There is the text that updates for the timer and beacons progress in which there are 4 of, then there is the progress bars made by manipulating quads. I create the progress bars by layering up 3 different quads, there is a background, middle and foreground and my initial idea was to be able to have a texture for the middle then lay over a foreground colour with alpha to give a sort of greyed out effect with the full colour part expanding as the bar progresses. All the class requires in its constructor is the dimensions, placement and textures and it handles the rest also having an update function to change the percentage. It does support loading bitmap textures for each part of the bar however in the final build I decided to go with flat colours on all of the bars as a stylistic choice.







I modified the quad class by adding a percentage and increment variable, using these two of the x coordinates can have their location changed accordingly to whatever percentage is given in the update function. One thing I would do if I had more time with it would be to add interpolation to the x values so instead of them jumping up it would be smoother.

# Part 2 – Camera, Meshes, Lighting, FX

## Camera Motion

With the genre of my game being an FPS I decided to attach my camera to a player game object and position it about head height, the camera follows the object in a similar way to the third person camera but with the ability to look wherever the player wants to still. I also attached a mesh to the camera itself for the players weapon using the a slightly modified version of the code given in the FX lecture. While its not directly part of the camera, the player projectiles also use the same positioning code as the gun as their spawn point. I added a modifier to the camera class allowing for redefinition of the FOV, using interpolation there is a smooth transition between the regular FOV and the one for sprinting.

There is a secondary view mode as well which is when placing a camera, this feature was created for when my initial idea was a stealth game however I transitioned into a wave shooter so while it isn’t very useful in the gameplay it is still a feature in the game. You can place 3 cameras and view through them at any time cycling through the placed cameras with period or comma.

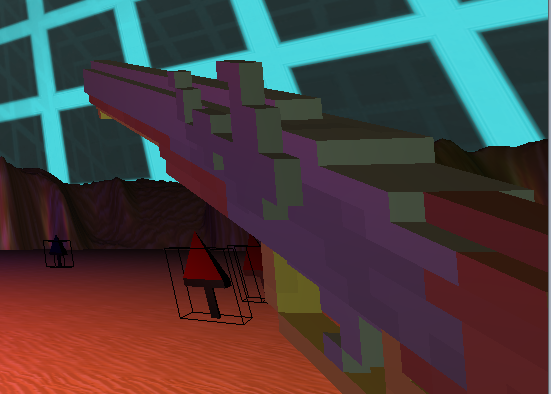
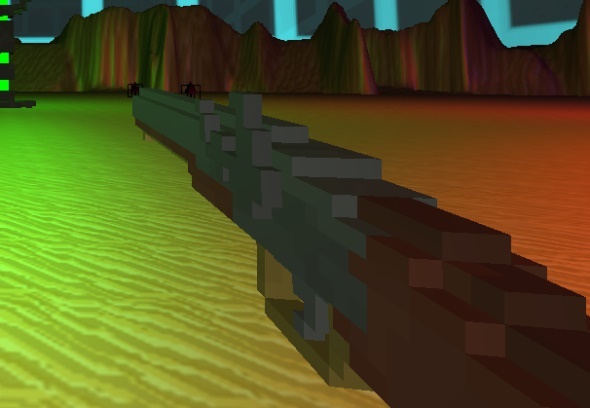
## Mesh Objects

There are a lot of meshes in this game, 16 in total as the menu screen alone makes up 9 of them. Rather than highlighting how every mesh could fit the requirements I will pick out 3 that are all implemented very differently but fit the requirements.

The first is the flintlock pistol, this is a game object that is only loaded in and doesn’t really have a physics affect on anything else, it gets translated into a position so its always in view using the camera right and front vectors, it gets rotated depending on the angle the player is looking and then it also gets scaled to fit into the screen. The lighting behaves normally on the model and there is even a spotlight behind the player specifically to make sure the model is always looking how it should and not too dark.

The next model is enemy crabs, there are 4 individual models with very slight differences, but the ranged crab mesh is where the most manipulation takes place. The crab is scaled to look right in the game world. The translation and rotation of the crab is dependant on its actual position which is a determined by the AI and physics manager. When the ranged crab is within a certain range it will aim at the player.

The last example is the beacons, these are scaled to a much larger degree than any of the other game objects in the project, they have their own hitboxes so you cannot walk through them and all have different positions. The SoRT of the beacons is more basic than the rest of the game objects as they are static however they still fit all of the requirements. Each one has a point light inside them which lights the mesh up with the beacons distinct colour.



## Lighting and colouring effects

To fit with the theme of the game I removed the directional light and instead have 3 different coloured point lights for each of the beacons in the game. This presented challenges as I spent way too long trying to get it to work before remembering there was a limit on point lights in the shader so I changed that to 3 for the look to work. This has a negative impact on performance however I think it is worth it to complete the look. As there was no directional light the menu screen was unseeable so I added a point light there that is only rendered during the menu screen too so that shouldn’t effect the performance when in the level.

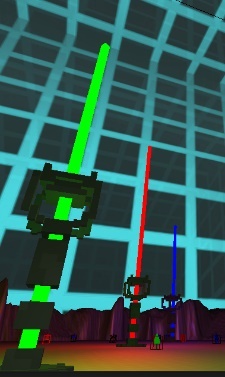
For my moving light I used a spotlight and placed it behind the player, it is just a white light to make sure the players weapon is always lit up correctly. It is position at the players position minus the front vector but point towards the front vector so its always pointing where the player is looking.

## Special Effects

The special effects were what I was most unsure with for this project, I have an explosion for when enemies die using code from the FX lab but other than that my other FX is a bit less obvious. When you place a camera and view it there is a cut out effect so you only see a circle and I think that would count towards FX. The beacons beams themselves all constantly rotate and grow as their percentage increases and I think that should also count as a special effect. The last thing that could potentially be seen as a visual effect is the FOV change as you sprint, I was thinking of adding a crossfade as well but I think the change in FOV is enough.

A screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a video game

Description automatically generated

# Part 3 – Physics, AI, Gameplay

## Game physics

In this project I spent a lot of time going back and forth with how I would move the player, at first I purely updated the position and tried my own collision code to stop the player from moving through objects but all it would do is slow them down instead of a dead stop. I then tried to use the bullet physics instead and had to add velocity but it didn’t quite give the feel I wanted with the movement even though collisions worked. I ended up using just velocity for the movement instead of changing the positioning as well and with tweaking got the feel I wanted.

Another part of the movement was the sprint, I used an equation to accelerate and decelerate at a non linear rate when the player goes in and out of a sprint to give a good feel to the game.

Text

Description automatically generated

All the game objects in the game other the flintlock and turret are linked to the physics manager and give resistance when walked into by the player and the AI. All these objects have mass and react in a convincing way as not to break immersion. There are a few moments where the AI makes a weird rotation, but I would say it does not deter from the experience. The menu screen as well shows a good use of physics with the masses of each of the objects set in a way to give a pretty creative and fun effect having the title letters being thrown around.

## AI

I think the AI is where I put the most research and effort into, I went with behaviour trees as I had an idea for one of the enemies that would require quite a lot of different actions decisions and paths. Before this I didn’t know much about behaviour trees so did more research past what was in the lecture. The two videos that helped me the most were by AI and Games (Behaviour Trees: The Cornerstone of Modern Game AI | AI 101, 2019) and a video lecture by [Petter Ögren](https://www.youtube.com/channel/UCtmNf5go0Xu81l3YT9xF1LA) (What is a Behavior Tree and How do they work? (BT intro part 1), 2019). With that I was then able to create some diagrams for how my behaviour trees would look.

There would be two trees, one for the engineer enemy and then one for all of the rest, to fully understand my design choices I think its best to explain my intention for the engineer enemy. The idea for the engineers is that they would work in parallel against the player. They would never approach them instead run around and undo the work the player had done turning on the switches, placing down defences. If the player got too close they would run away whilst placing turrets to distract the player.

Diagram

Description automatically generated

This website I used to create these trees (<https://www.behaviortrees.com/#/dash/home>) only allowed to create them horizontally so work from bottom to top in terms of decisions. I wasn’t too sure at this stage how I would approach the programming so for simplicity sake I made sure each branch only ended with one executable node as that made more sense in terms of implementation later.

The second tree I came up with could have been made in a basic state machine but as I was going to implement behaviour trees anyway, so I decided to go with a tree. The only difference between the melee enemies and the shoot enemies will be the range values in which they can engage the behaviour otherwise is the same.

Graphical user interface, diagram, application

Description automatically generated

With my basic understanding of behaviour trees there I went about researching how to implement them and stumbled across a guide by a programmer going by the name prestokeys, they create a guide on the forum cplusplus on how to implement them (Behaviour Tree, 2021). They provided templates for nodes, composite nodes, sequence and selectors as well as a node they called action. The post went through another blog on the website game developer explaining behaviour trees and coded one of the examples there. (Simpson, 2021).

I did some tests and used their code for the nodes within my own getting the trees working with just strings first, after this is where my own adaptation came in as I redesigned the action node to take in my state enums and a variable to keep track of what the last executed node was.

A picture containing text

Description automatically generated

Text

Description automatically generated

This changed allowed me to separate my actions into two types, decisions and executables. The decision actions would pass through a bool that changes depending on certain values like range from player or if there are any switches on while the executables would always pass through true. The last\_state let me keep track of where the tree ended and what state to choose.

A screenshot of a computer

Description automatically generated

This let me update the state on every call of update() and use a state machine to then call the appropriate function. This leads onto the next part of the AI, after the decisions were made, I needed some functions to actually call and perform those behaviours. Most of the movement functions were very simple, such as with fleeing they would just move the opposite of whatever the direction vector towards the player was. To patrol I generated a random point and once within a close proximity generated another so the AI would keep wondering until the state changed. While the movement functions, themselves were rather simple I kept running into an issue where the enemies would get stuck on objects such as switches, the beacons or each other. To fix this I added some more states and checks so that if the enemies was in their approaching or patrolling state and hadn’t moved a certain distance in the last few seconds they would generate a new random point a short distance away, once they reached that point they would go back to whatever the previous state was and continue on their path. If they still hadn’t moved again another point would be generated and this would keep happening until they became unstuck. In my testing I found this solution worked rather well. The enemies may have no flocking but the unstick procedure seemed to circumnavigate that issue.

## Gameplay Elements

I wasn’t too sure on exactly what the brief wanted in terms of gameplay elements, so I created the mechanics I wanted for the game first then added on a few extra bits to satisfy the mark scheme. In terms of what I added to fix the mark scheme there are two powerups, these have a chance of spawning every 5 seconds and can either be a health pack or a beacon boost. The beacon boosts increase the speed of the beams incrementally and they’re permanent stackable boosts. There is also a timer that counts down and if all the beacons aren’t at 100% before that timer is done you lose, the timer has the exact amount for the chosen song to come to an end as you lose.

In terms of what I wanted for my gameplay elements there are the beacons and switches. The beacon class is created with a few variables the key ones being how many switches, initial beam speed and how many switches need to be activated for the beams to be active. For the sake of simplicity in the demo I gave each beacon 6 switches and only 4 need to be active per beacon for them to work. Each switch has its position randomly generated in a radius around the beacon so they’re all in a similar distance from their respective centres. When a beacon is active the beam gets higher in the sky and its percentage goes up, the game manager takes an average of the three beacons, and this is the level progress bar at the top of the game. Only when all three are at 100% you win. As its currently implemented once a beam is at 100% it can go back down again but I think I would possibly change this so once its at 100% it locks in. However this does add an extra element of chaos and speed going across the level constantly to juggle the switches of all the beacons at once.

## Variable Difficulty

There is not mention in the mark scheme but on the brief it mentions varying difficulty, for this I have the potential enemy spawns tied to whatever the overall percentage of the level is to being complete. It is split into 20% brackets and every 5 seconds a function is called to spawn enemies depending on what bracket the player is currently in. It starts off easy with just melee and ranged enemies spawning but as time goes on more engineers spawn and eventually shotgun enemies too who deal more damage. In each bracket the spawns aren’t the same every time as there is a random chance of what will spawn in each bracket. The only issue with this system is its been quite difficult to playtest and balance with the time I had left so it is still unclear if the game is beatable or too difficult in its current state.

# Reflection

Overall looking at the game I have managed to make with the time given I am quite pleased; I feel like if I had more time there would be a lot of things I’d like to change but I think I made a good attempt at using all the tools and techniques given to us in a constructive way to benefit my project. I feel like as it currently stands bugs and optimisations aside it is a very playable demo for what could be turned into an actual game.

The parts I am most happy with are the AI and overall game logic, being able to put tons of enemies on screen and have them behave in a convincing way is my favourite part of the project. I think another strength of the project is the visuals, the lighting works well with the environment and the there is a semi consistent style to the meshes used and look of the game. The UI works almost exactly how I wanted it to and presents the important data the player needs without taking up too much of the screen. Another part I think is a strength although its not important compared to all the other elements is the menu, Its quite fun having all the letter be thrown about and it’s a good introduction to all the enemy types that are within the game.

There are more aspects of the game I am unhappy with and would want time to improve on though, optimisation is the first. When writing this report and getting screenshots I couldn’t even run the game on my laptop correctly the physics was too slow and all my game objects fell through the world. Now my laptop isn’t the most powerful however the game shouldn’t be that intensive either. The spawning system is very demanding and causes large performance dips so I would try and come up with a better system than that if I was to spend more time on it.

There are a lot of bugs in the current project such as the hitboxes of enemies getting flipped upside down but the enemies staying upright meaning the projectiles could occasionally go through them and they’d be harder to kill, the projectiles themselves also have their own issues as they don’t go straight towards the centre of the screen so it can be hard to aim them when first playing the game.

**What would be required to expand this into a complete game**

I think there would need to be more cohesion between the collision systems I use, I use a mix of the bullet physics for movement and regular bounding box for projectile collision I think for performance and less bugs using one system for everything would be better.

Adding flocking to the AI as well would be useful as right now if a load of them come towards you at once its more like a massive blob of enemies than a wave of them, this would also lead into proper obstacle avoidance. For this to be a full game I think more props and a better-looking world would be required and while the current avoidance code where they get themselves unstuck works, in a level with a lot more props having proper obstacle avoidance would make the experience a lot smoother.

The movement in the game could do with a lot of work. I would want to add sliding, better momentum mechanics, possibly a working jump and support for controllers or any other input device. Other than these mechanical changes more content of course would be required, maybe even a story mode for this kind of shooter and this demo would just be the tutorial.

# References

*Behaviour Trees: The Cornerstone of Modern Game AI | AI 101*. 2019. [video] https://www.youtube.com/watch?v=6VBCXvfNlCM: Youtube.

*What is a Behavior Tree and How do they work? (BT intro part 1)*. 2019. [video] Directed by P. Ögren. https://www.youtube.com/watch?v=DCZJUvTQV5Q: Youtube.

*cplusplus*, 2021. Behaviour Tree. Available at: <http://www.cplusplus.com/forum/general/141582/> [Accessed 2 December 2021].

Simpson, C., 2021. Behavior trees for AI: How they work. [Blog] *gamedeveloper.com*, Available at: <https://www.gamedeveloper.com/programming/behavior-trees-for-ai-how-they-work> [Accessed 1 December 2021].