Minor Skilled

Tank Simulator

Term 3 & 4

Nils Meijer – 466301

Table of Contents

[Product description 2](#_Toc137989664)

[Project aspects I worked on: 2](#_Toc137989665)

[Movement system 2](#_Toc137989666)

[Camera controllers 3](#_Toc137989667)

[Turret controller 4](#_Toc137989668)

[Shoot system 4](#_Toc137989669)

[Level design 4](#_Toc137989670)

[AI behaviour tree 4](#_Toc137989671)

[Damage registration system 4](#_Toc137989672)

[Main menu 5](#_Toc137989673)

[New input system 5](#_Toc137989674)

[Polishing (adding “juice”, post-processing), fixing bugs 5](#_Toc137989675)

[Progress comparisons 6](#_Toc137989676)

[Workflow 6](#_Toc137989677)

[Hour registration & planning 6](#_Toc137989678)

[Quality Assurance 7](#_Toc137989679)

[Visualizations & diagrams 7](#_Toc137989680)

[Future features & iterations 7](#_Toc137989681)

[Appendices 8](#_Toc137989682)

[References 8](#_Toc137989683)

# Product description

For the past 5 months, I have worked on a tank game. The original idea was to develop a simulator, meaning the experience would be close to reality and how tanks operate in real life. However, this eventually evolved into a less realistic tank “game”. Because of time constraints and difficulties in developing certain components of the project, I couldn’t fully complete the idea I had while brainstorming in the first few weeks. In hindsight, it would have been better to have brainstormed a slightly smaller, less ambitious project so that I would have more time to focus on research and properly test & implement the features.

In the final product, the game starts in an interactive main menu, with an option to quit the game or to proceed to the next scene.

## Project aspects I worked on:

* Movement system
* Camera controllers
* Turret controller
* Shoot system
* Level design
* AI behaviour tree
* Damage registration system
* Polishing (adding “juice”, post-processing), fixing bugs

### Movement system



Figure Tank tracks example (source: <https://edition.cnn.com/2023/03/21/politics/us-abram-tanks-accelerate-ukraine/index.html>)

I started out developing this system with the ambition to simulate real-life tank tracks as closely as possible. With a special type of colliders in Unity, which are WheelColliders, I would be able to create a tank track similar to the example in Figure 1. The wheels inside the track would respond to terrain changes, due to the suspension system. The only thing different would be that the WheelColliders would also be the contact point to the ground, rather than the track. In a real tank, the front and rear sprockets are ”powering” the track, which then makes contact with the ground. Due to that friction, the tank can move.

However, during a large part of the development, I struggled with making the WheelColliders behave the way I expected them to. The tank wasn’t even able to rotate properly. The way tanks handle rotation is to move the track opposite the tank is trying to move towards. Example: if the tank should rotate clockwise (so to the right), then the track on the left should be activated. Optionally, the track on the right can be reversely activated to create a stronger rotation force. This is how I tried it with the WheelCollider approach as well, but it didn’t work as expected. What happened was that it would either barely move at all, or the tank would need to travel a very large distance forward before having rotated the desired angle on the Y-axis.

Another reason why I struggled with implementing this WheelCollider approach was that, no matter how much research I did, I couldn’t wrap my head around the WheelCollider parameters that could be customized. Think of variables such as “Extrenum Slip” and “Extrenum Value”, and how they influenced eachother. I spent too much time trying to find a balance in these values.s

I decided to greatly simplify the whole movement component, and just resort to just adding forces to the Rigidbody of the tank.

### Camera controllers

As I was doing my research, I noticed many games had different (or multiple) camera views. In the beginning, I had 3 different views, which would be:

* **3rd person**



Figure Camera gizmos

The camera follows the tank at a set distance, and can rotate a full 360° around the tank, as well as a limited range of motion on the x-axis. This will likely be the camera view that players use the most, to navigate around the scene and shoot targets that are relatively nearby. For this camera view, I did my best to visualize how the camera worked, by showing the boundaries for movement, and how it relates to the barrel.

* **1st person**

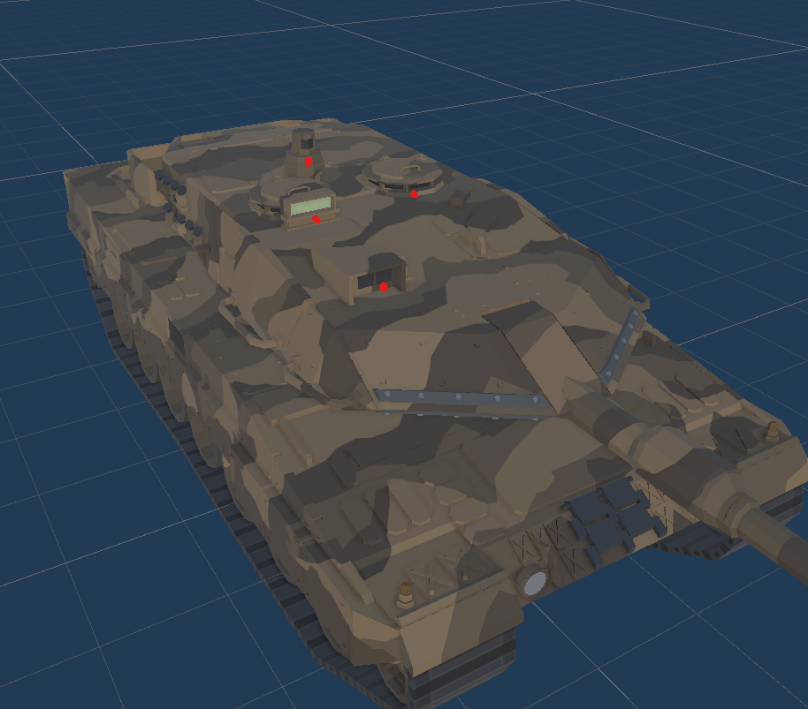


Figure , 1st person camera positions (red dots)

A camera placed on any of the 4 positions marked in Figure 2, unable to move by itself. It was an attempt to mimic reality, but it wasn’t a gameplay element I had any use for. That’s why I decided to remove it, improving gameplay quality by decreasing complexity, at the cost of decreasing realism.

* **ADS**

A camera placed right above the barrel, which follows the y-rotation of the turret, and the x-rotation of the barrel. The player is able to zoom in using the right mouse button up to 5 times. This is done by decreasing the FOV, rather than physically moving the camera to a position in the distance. This would have introduced issues, such as moving into walls, being able to see behind walls and other edge cases that could not be solved which is why I changed it to FOV adjustments.

This camera view should be used to target enemies at larger distances, since the zooming ability helps with hitting those targets with high accuracy.

### Turret controller

A system to control the rotation of the turret and barrel of any tank this component is attached to. The player can do this by rotating the correct input axis (mouse, joystick). This will instantly rotate the “target rotation transform”. The turret uses this as a target to which it will rotate towards.

### Shoot system

Developing this component was another way to try and make the game a bit closer to reality. When using the proper input, a shell is spawned at and shot from the tip of the barrel.

During early/mid development of this feature, I spent a lot of time on a “ranging” system. During the research phase of my minor, I found that a lot of games had some kind of range finding way. For example, you would manually measure the distance to a point using UI called “mildots” (or the distance was measured automatically), and you would have to adjust the rotation of the barrel on the x-axis to account for shell drop, making sure that the shell would hit the target in the distance accurately.

Unfortunately, this didn’t work out.

### Level design

During programming, I sometimes need a break to replenish my energy for it. But to stay productive and spend my time on something useful for the project, I created a small level in which the player can move around in. It’s not finished by any means, but it’s good enough for a prototype level.

### AI behaviour tree

To blow more life into a game, in my opinion, it needs a presence other than the player itself. That’s why I decided to add AI agents. They are able to move around the world and make their decisions by the use of a behaviour tree.

### Damage registration system

In a tank combat game, the tanks obviously need to be able to be destroyed. That’s why I spent a few weeks on developing a damage registration/health & armor system, as well as a way to

### Main menu

Every game of course needs a main menu to start the gameplay. I’ve always been a fan of these interactive menus. It makes the game feel more alive and adds to the atmosphere. For example, in my menu, I added a tank of which the barrel follows the position of the mouse. When I click any of the buttons, it shoots a shell, playing an explosion particle system. When loading the main menu scene, the gameplay scene is loaded asynchronously in the background. This prevents any loading screen or hickups/lag when clicking “play”.



Figure My main menu

### New input system

At first, I was still using the input handling system I was used to. However, during one of the Guided Work meetings and one of my QA sessions, I received the feedback I might want to check out the new input system, designed with several input devices in mind. This way, it’s easier to add new ways of input later on in development. It took some time and getting used to the new approach of handling input, but I got at least the basics of how it works down. Overall, it’s more robust and efficient.

### Polishing (adding “juice”, post-processing), fixing bugs

Added FOV animated change (higher FOV when driving, normal FOV when stationary) to 3rd person camera.

# Progress comparisons

|  |  |  |  |
| --- | --- | --- | --- |
| Evaluation 1 | Evaluation 2 | Evaluation 3 | Final presentation |
| [Tank simulator demo - 16-03-23](https://www.youtube.com/embed/MZfnDB9sM3g?feature=oembed) | [Tank simulator demo - 20-04-2023](https://www.youtube.com/embed/_fXscsjpiMA?feature=oembed) | [Tank sim demo 30-05-2023](https://www.youtube.com/embed/HcJr45zD4Rw?feature=oembed) |  |

# Workflow

I did all my hour planning, brainstorming, functionality planning and documenting in a software called “Obsidian”. I hadn’t used it in a (for me) large scale project before, so I wanted to see how it would affect my workflow for such a long-term project. There are some things that can be improved, but overall I liked working with it.

During the first few weeks, I would go to school when I knew some of my friends were there as well, so I’d force myself to get out of the house more and make sure I’d get some social interaction. However, apparently it’s required to “reserve” the monitors/PCs in the XR lab and I’d often not know if I would be at school. There were a few occasions I managed to get a 2nd screen to work on, but often I just had to work on my small 17” laptop screen. So after probably the 1st evaluation, I really only came to school for the Guided Work meetings and just stayed at home the rest of the week so I could work more comfortably at my own PC. I’d often sit in a Discord voice channel with some friends to get some social interaction.

Every day, I would work from ~8.30-9.00 to around 16.00 so that equates to ~35 hours (5 \* 7) a week. I usually keep working during lunch. In the last quartile however, I struggled with keeping up motivation and hours (together with combining working on redos, going to the gym, working at my sidejob, keeping my place clean and seeing my parents during the weekends) so those hours are more in the range of 30-32 hours.

# Hour registration & planning

I tracked my hours using the software “Toggl Track”. The program is basically just a stopwatch, which remembers the time entries so it’s easy to see how much you worked every day and on what aspects of the project. In the desk view, it’s possible to export a visualization of this data, which is what I used during the evaluation presentations and the final presentation to justify my hour registration.

Around every evaluation, I would look at my hour planning and check where it needed readjusting. During my internship, I also had to estimate how many hours I would need for a certain task, and review that planning after completion (and also update it during development, of course). That experience helped with doing the same for this minor. But as with all things, my hour estimation skills can still be improved (and will happen as I grow as a programmer).

# Quality Assurance

Before any QA meetings, I was planning on doing a QA with professional programmer in the game development industry. For the first meeting, I messaged and emailed several studios working on a tank-related game. I also made posts in several Reddit game development groups. Unfortunately, I got 0 responses (and in the reddit groups, I even got downvoted). I ended up asking a former colleague from my internship at Total Reality as the first Quality Assurer. He replaced the previous lead developer (who was my internship coach), but left the company again after about 2 months. For the 2nd, 3rd and 4th Quality Assurance meetings, I asked Hans Wichman and Yvens Serpa.

I also made a lot of use of ChatGPT during development, although it doesn’t count as QA in my opinion. Rather as a way to help me get out of being stuck with a certain problem.

# Visualizations & diagrams

I made visual explanations for many parts of my project. One example is the behaviour tree diagram.

# Future features & iterations

* Tank customization
  + Originally a feature I was planning to develop for this prototype. I wanted the player to have more control of the tank they were playing with. Both in a gameplay aspect and a visual, eye-appealing aspect. For example, purchasing a light source that can be attached to the tank to make it easier to navigate through a level when it’s dark. This would be a gameplay customization. A visual customization would be buying a different skin for the tank (another camouflage colour).
* More sound effects
* Placing landmines
  + This would improve the combat dynamics, as both the player and the AI can decide to place landmines in a strategic location, which enables more complex combat scenarios.
* A dynamic day-night cycle.
* Other types of ammo
  + Having several types of ammo (which was the plan at first but has been scrapped due to time constraints) would make the combat more dynamic, forcing the player to choose about the way they approach a combat scenario. For example, if you inspect an enemy tank and see it has exceptionally high armor properties but relatively low health properties, you would want to choose an armor-piercing shell round so it’s a way around the heavy armor.
* More levels
  + Combined with the quest system, this would make the game more immersive, because there could be a story-telling aspect if there were several levels.
* Quest system
  + A feature that allows for an actual “purpose” in the game, rather than just driving around and killing enemies.

# Appendices

# References