Tobii Maze

New Technologies - Report 2

Erik Oussoren
Ananya Ghosh
Pepijn Thijssens
Felix Buenen

Project Summary

In this project we implemented a simple top down maze game. The player is responsible for getting a robot out of the maze. However, the robot is not very clever: it walks in a straight line towards the gaze position of the player. This way, the player has to lead the robot out of the maze.

Motivation

Conceptual
We found it interesting that by simply looking in one direction, the character moves accordingly. So, if you look in the wrong direction while trying to figure out the way in the maze, your character moves in the wrong direction thus, you have to direct your character in the right direction strategically.

Technical
We used the Tobii EyeX because it is an easily accessible tool, easy to set up and its SDK is easy to get started with.

Decisions & Choices

Conceptual
Since it is quite hard to stare away from your moving character we decided to add some background dots on the map to focus on to make the movement easier on your eyes.

Technical
The character is programmed to go directly to the player's gaze position. That is, we did not include any form of pathfinding. Initially we thought of including pathfinding because it looks better, however, that would mean that the player could just look at the finish and the character would 'pathfind' his way out of the maze… not a very interesting game! On top of that, the game can actually become interesting if the player 'uses' the dumb behaviour of the robot. For example, the player can first lead the robot towards a corner. Then, the player is safe to analyze

the maze in its entirety, because the robot will just continue to walk into the wall; it will be locked in place.

This decision made our Tobii Maze game fairly straightforward in terms of technical difficulty. The character's position gets updated every frame with his velocity. The velocity is constantly updated with the following formula:

```
velocity = gazePosition - robotPosition;
```

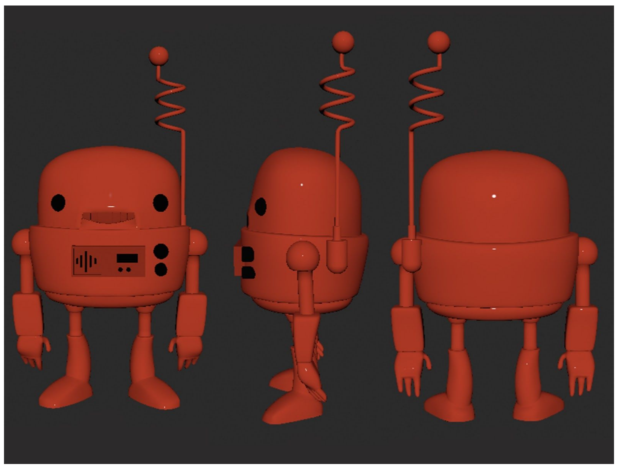
This creates a vector that points from the character position towards the player's gaze position. We normalize this vector, multiply it by a variable speed factor and we have the character movement covered.

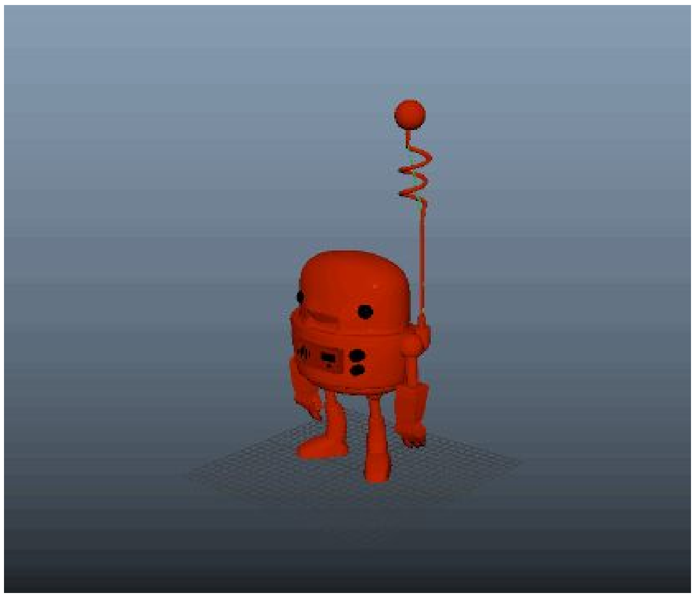
The only thing that is left is the collision detection. Unity provides this functionality for us. The only thing we had to do was making our character and maze walls rigid body objects.

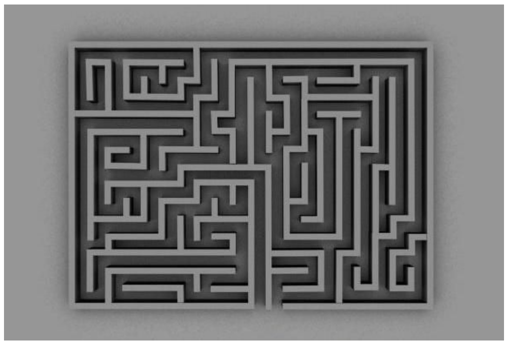
For the art we decided we wanted to use a little robot as the character following the player's gaze. We gave the robot an antenna, which would act as a clear indication of the direction the robot is moving or trying to move.
The antenna would basically be pulled towards the player's gaze if the player isn't looking directly at the robot itself. In this way, the visuals were supposed to aid the gameplay.

The implementing of the model of the robot went wrong however, and we didn't have time to fix it in time. Something in the rig's weight painting didn't export correctly, resulting in polygons lacking behind others in the animation.
The antenna did work fine, but it also was a separate object. Part of model was separated in parts, mainly the arms. They were parented to the rig, and also didn't seem to be a problem. The body did have some problems however, which made the animation pretty unusable in Unity. I've tried multiple different export options in Maya, but they didn't help at all.

Gallery







Link to the video:
https://drive.google.com/file/d/1evl_7TP_MjWoV1XlCg5PhXClp3J3i_T_/view?usp=sharing

Team contributions

Who?	**What?**
Erik Oussoren	• Rigging & Animation
Ananya Ghosh	• Robot & Maze Model - Maya • Final Presentation
Pepijn Thijssens	• Game Dev - Unity • Documentation
Felix Buenen	• Coding & Game Dev - Unity • Documentation