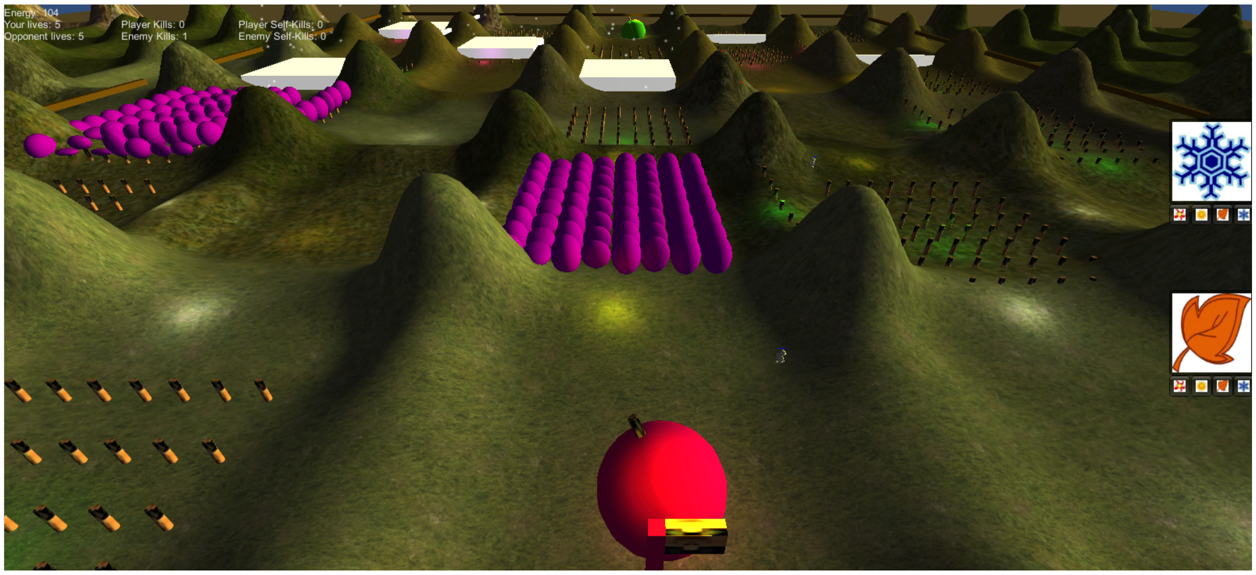
Starving Worms,
Crawling to Victory

Cléa Benz, Nicolas Imhof & Adrien de Gottrau

(A big thank you to Jan Wolf, who sadly couldn't find enough time in his schedule to help us until the end of the project.)

Interim Report



A current game scene

Status Layered Requirements

In the following we show the status of our game. As you can see, the functional minimum has already been fulfilled. The low target is nearly complete as well. The only open point is the "simple animations" which means that our game is currently missing animated changes of the season, trees just pop up and disappear instead of growing and decaying.

This delay can be explained/excused by the difficulty of the asset generation with the overloaded Maya program.
But besides this little issue, our game is already in a quite advanced state and can already be played against the artificial intelligence which is already challenging for a beginner.

Legend : green = done, orange = in progress

Functional minimum :
- one simple map (with grid structure)
- one fruit (cherry)
- worm spawning and walking (shortest path) to the opponent’s fruit
- terrain elements: forests (that react to the seasons), mountains
- primitive graphics
- player interface (information display: seasons, energy)
- ability to change the seasons

Low target :
- simple animations
- singleplayer (primitive AI)
- ability to switch the terrain of the sections
- more terrain elements: water, mountain paths
- game menu

Desirable target :
- two worm clans
- advanced graphics
- advanced AI
- small tutorial
- good game balance
- game presentation video

High target :
- multiplayer
- additional maps
- very good game balance
- intuitive user interface
- map generator
- ingame intro (video)

Extras:
- worms fight each other
- worms evolving

- event at each season start
- interactive tutorial
- campaign
- dynamic weather system
- chat system
- many terrain elements
- towers
- extra terrain elements, extra worm types
- fog of war

Implementation

The code is very well structured, object-oriented and easy extendible. This will surely serve as a big advantage in the future since the programming progress will not slow down because of bugs or a missing overview about the code.

In the following some classes are presented to give a basic overview about the code.

MenuBehaviour
This class handles the game menu. The first thing a new player sees of the game.

Game
The game class is the root of the whole game. It contains references to almost all things that need to be accessed somehow. It initializes the game map, the players and the user interface.

Season
The season class is used to simulate a season in the game. There are currently always two seasons in the game which divide the game map in two parts.

GameMap
The game map gets initialized by the game. It contains the playable map structure, the map elements, the position of the player's fruits and can be arbitrarily scaled.

MapElement
A MapElement is a border or a tile. Borders are vertical or horizontal and just block the worms' path from one tile to another when they are active. There are several types and each type is only active in one season. E.g. trees only block in the spring season.

Player
The player class that simulates a player. Contains all information about the player, it’s fruit, energy and also contains a reference to an artificial intelligence if not controlled by a human player.

Artificial Intelligence
The artificial intelligence (AI) is our technical core focus, which is why we already spent a reasonable amount of time to it. An AI also provides the advantage that you can play the game a little, which has a motivating effect and it also provides a better feeling of for the game. There is a separate folder for the AI. It contains the basic AI class which is used to extend several AI types. Currently there are four simple AI implementations.

1. The RandomAI simply does random actions. (changing a season to another, changing a border or a tile) It does not at all react to the game state in any intelligent way and was the first AI that was implemented.
2. The KillerAI performs already quite good. It simply tries to kill the worm that is closest to its fruit by changing the season. If it is not possible for the AI kill the closest worm, it behaves like the RandomAI. Still, it tries to save some energy so that it is ready to kill when the closest worm gets on a dangerous tile.
3. The RunnerAI tries to free the way for its own worm that is closest to the target fruit. It therefore changes the tiles or the borders. If the AI’s energy exceeds a certain threshold, it does some random actions like the RandomAI.
4. The KillerRunnerAI is a simple mix of the two AIs but with priority on killing.

In a next step, an advanced AI needs to be implemented. This AI should check all its possible actions, rate them and choose the best one. Therefore it needs to be able to simulate a change of the seasons, a border or a tile. For this we also need to be able to access the worms walking paths. Currently we use unity’s NavMeshAgent to handle the worms walking and navigation and unity’s dynamic NavMeshObstacles to block the inaccessible terrain parts. An own implementation would probably be more efficient since we know exactly which parts of the map are when walkable and when not.

UserInterface
This class is used to handle all the UI drawing on the screen. This is very convenient because otherwise there would be calls to draw on the screen everywhere spread in the code.

AudioManager

This class is simply responsible for loading and playing sounds.

MyController
We used this class for the basic camera control. It will probably be replaced by a new class "DeviceController" to allow the control of the camera on a handheld device.

Unit
This is the class for the worms. Since it is not yet specific to a worm it could also be used to simulate other units. We first expected the pathfinding of the units in the game to be a big challenged and were surprised when we figured out that unity3d already provides a simple solution to our problem.

Coming implementation challenges
Currently we are facing performance issues on the tablet. We are not yet sure where those issues come from. Probably they can be solved without impact on the game. Otherwise we will need to tune some parts of the game.
We will also have to smoothly integrate the assets and their animations into the game.

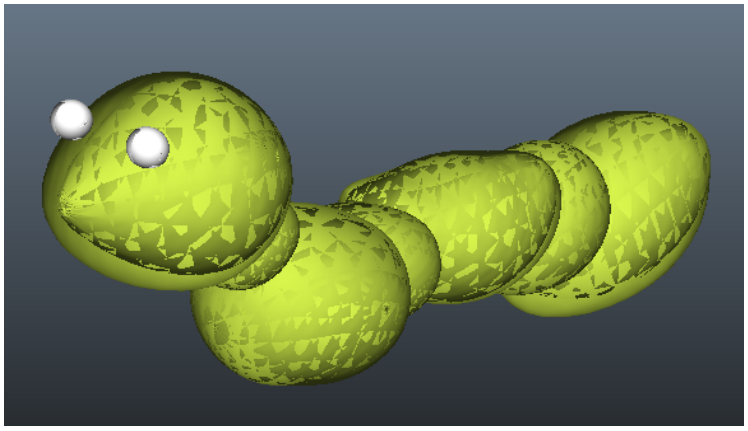
Assets & Maya

In parallel to the development of the game, we started to design and create the different assets we need for our game. In order to design them, we are using different drawing and 3D modeling softwares. The main software that we use is Maya, but we also have to draw some textures and pictures on Gimp, Photoshop or Inkscape.

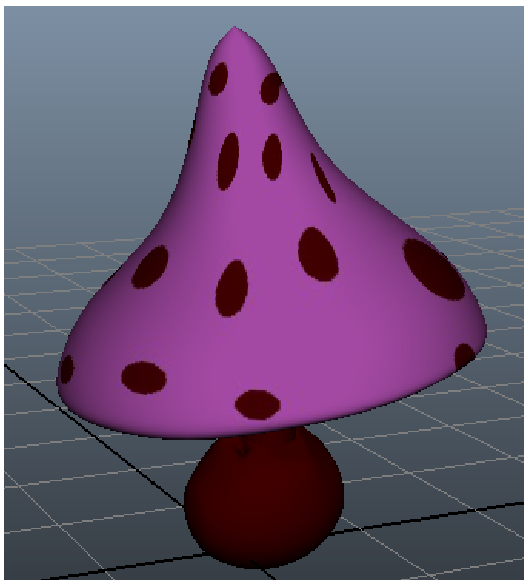
Maya is quite hard to master and it took us a long time to get into the program and to understand how it works and how to make good 3D assets that don't take too much computing power.

We made a list of the different assets we need for our game :
- Worm
- Forest
- Desert
- Snow
- Mushroom forest
- Apple
- Cherry
- Flag
- GUI Elements

We can see that we still have many assets to create and improve. The assets currently one of our main attention point and we hope to finish them until the Alpha release. Creating nice assets is really time consuming and we believe that we underestimated this aspect in our schedule.
We also lack some nice animations for our different assets. But once our assets are finished, we don't think that the animation will be a hard task. The main difficulty remains to keep the game nice looking and smooth running.



Worm



Some assets that we created

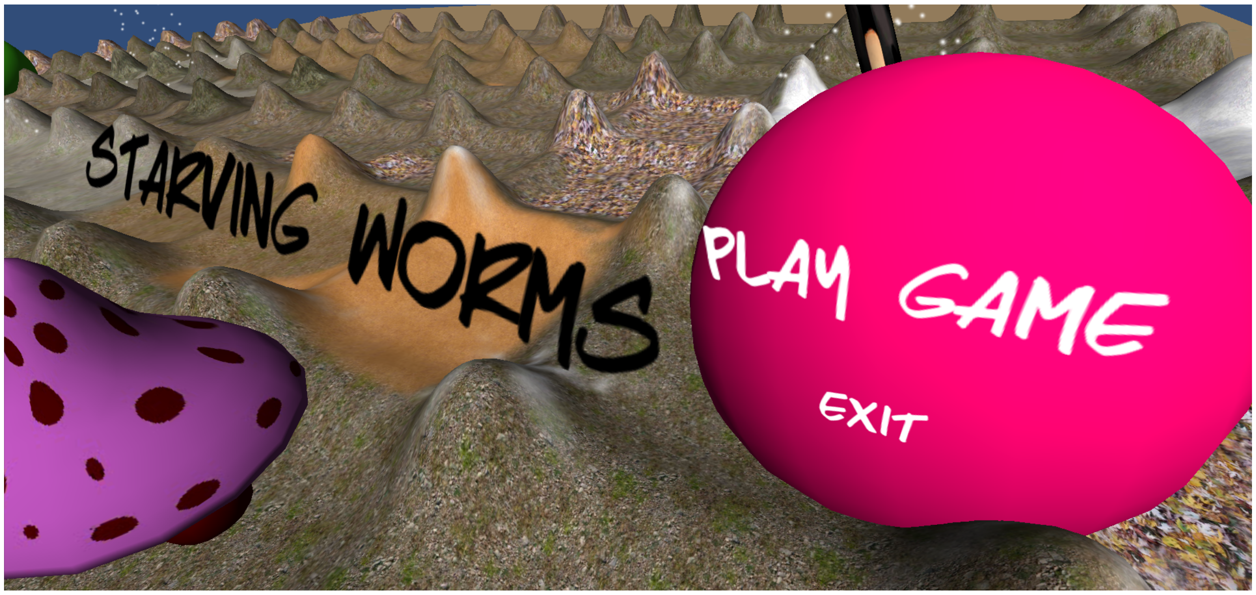
Sound & Menu

Sound

For the sound we oriented us according to the following youtube video: https://www.youtube.com/watch?v=GcNk1wV4eEc. The video explains what are the key points of game music. It should loop for ever, it is an addition to the game and not a distraction for the player and finally you should not get annoyed by the music.
To compose the music we used Mixcraft 6 and Audacity. We decided to use a background sound for the game itself and one for the menu. The music for the game is from the following website: http://ocremix.org/ and its called Kaleidoscope. If you switch the terrain or season successfully in the game a bell will ring, otherwise if you don't have enough energy or if there are currently some worms on the terrain an error sound is played. The music for the menu is build together in Mixcraft, the sound of the menu is rather sad, because the worms currently lost their favorite fruit. To keep the sound simple we only used some background drums and a piano melody in Gmol to express the sadness of the worms.

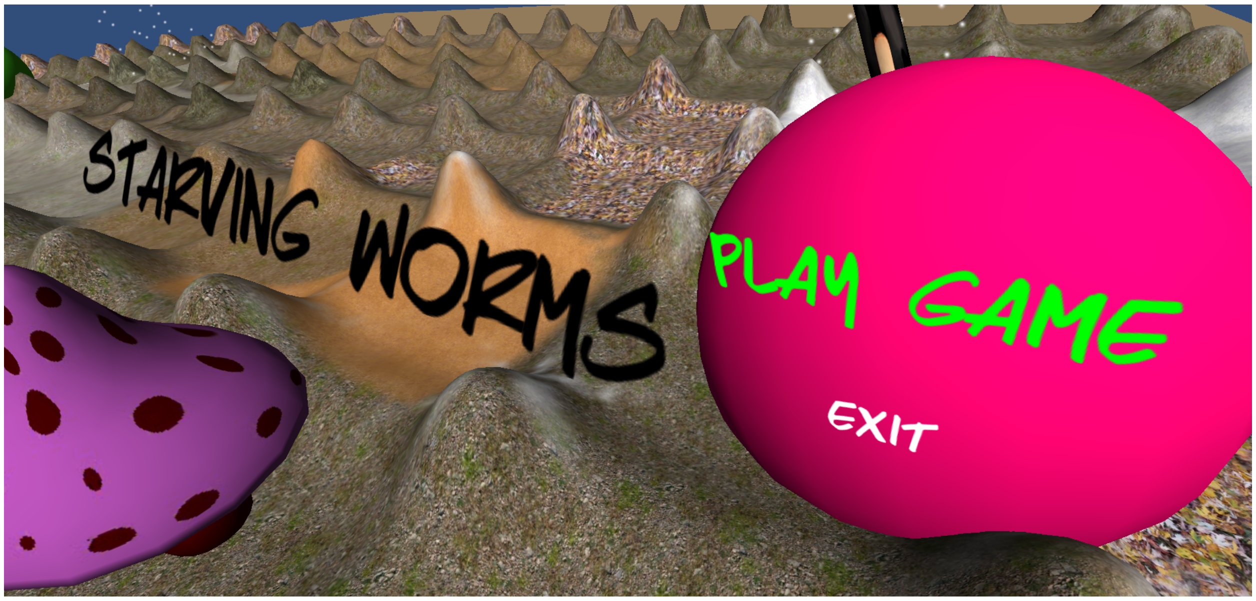
Menu

The menu should be inviting the user to play the game. Our plan is to use the scene of the game as background for the menu. To get a comic style look of the game menu we oversized the elements of the game. This is how our menu currently looks like:



The game menu

The Font is from the website dafont.com and it is free to use. The Font we chose is also in the comic style. If you drag your mouse over the play game button or exit button its color is turning green.



Highlighted “Play Game”

We still need to add a Settings Button and some more assets like a worm into the menu scene.