

Version 1.0

July 23, 2015

Presented by: jUstin morritt & Matt Manton

London

Contents

[Warganism 3](#_Toc425425388)

[Theme / Setting / genre 3](#_Toc425425389)

[Core GamePlay Mechanics (Brief) 3](#_Toc425425390)

[Controls 3](#_Toc425425391)

[Monetization Model 3](#_Toc425425392)

[Monetization type 3](#_Toc425425393)

[Project Scope 3](#_Toc425425394)

[Game Time Scale 3](#_Toc425425395)

[Team Size 4](#_Toc425425396)

[Licences/ Hardware/ Other Costs 4](#_Toc425425397)

[Influences 4](#_Toc425425398)

[Television 4](#_Toc425425399)

[Games 4](#_Toc425425400)

[Movies 4](#_Toc425425401)

[Target Market & target PlatForms 4](#_Toc425425402)

[The Elevator Pitch 4](#_Toc425425403)

[Target Contact Demographics 4](#_Toc425425404)

[Platforms 4](#_Toc425425405)

[What sets this project apart 5](#_Toc425425406)

[Core gameplay MECHANICS (detailed) 5](#_Toc425425407)

[Mechanic #1 – Strategically place different types of babies 5](#_Toc425425408)

[Mechanic #2 – Collect Zombie Brains For Currency 5](#_Toc425425409)

[Mechanic #3 – Advance to new Stages And Levels Of Higher Difficulty 5](#_Toc425425410)

[Mechanic #4- Babies Will Fight Zombies 5](#_Toc425425411)

[Story and gameplay 6](#_Toc425425412)

[Story (BRIEF) 6](#_Toc425425413)

[Story (Detailed) 6](#_Toc425425414)

[Gameplay 6](#_Toc425425415)

[Assets 6](#_Toc425425416)

[2D Textures 6](#_Toc425425417)

[Art Samples 7](#_Toc425425418)

[SOUND (Ambient) 7](#_Toc425425419)

[SOUND (PLAYER/COLLISION) 7](#_Toc425425420)

[CODE 7](#_Toc425425421)

[ANIMATION (Environment) 7](#_Toc425425422)

[ANIMATION (Character) 8](#_Toc425425423)

[Menu and Game Layouts 8](#_Toc425425424)

[Differnce Between a Baby and Zombie 9](#_Toc425425425)

# Warganism

## Theme / Setting / genre

Warganism is a top down, split screen, shooter that takes place in a limited area also known as a petri dish. The players will take on the role of a dynamically growing organism in a petri dish and thrive to survive against each other. The game takes on a minimalistic cartoony look that focuses more on the mechanics then the art. The genre of this game would be classified as a top down, casual shooter. Warganism may be played against a friend as Player vs Player or be played against our Artificially Intelligent organism.

## Core GamePlay Mechanics (Brief)

- Acceleration/ Deceleration with movement.

- Projectiles Emitted From Main 2 organisms.

- Pickups to help fend off the competitor organism.

- AI programmed organism.

- Player vs Player Capabilities.

- Dynamic Slime Tail that gives great visual feedback

### Controls

Player 1 will be limited to only using keyboard, the arrow keys to move and the space bar to shoot.

Player 2 will be limited to using the mouse to direct the organism around and left clicking to shoot.

The only other key that will be used in this game will be the P key, which is used to pause the game.

## Monetization Model

### Monetization type

This will be a free to play game, we plan on releasing it on the Green Light Steam Network.

## Project Scope

### Game Time Scale

We will be getting a little under 2 weeks total Coding Preparation. This includes everything from collecting all of the assets to programming it all into a functional game.

### Team Size

Team will consist of two people.

### Licences/ Hardware/ Other Costs

All assets will be using 100% open art. There will be no costs except for our time. As far as Hardware goes, this game should run on any windows computer.

## Influences

### Television

Bill Nye the science guy and his wonky organism cartoons.

### Games

This game has elements from the newly popular “Agar.io” which also has to do with organisms.

Other elements of Warganism may relate to the well-known “Binding of Isaac - rebirth”

## Target Market & target PlatForms

### The Elevator Pitch

Ever wondered the struggle of a living amoeba? Want to jump in the skin of one and defend yourself against your friends? Enter Warganism where you take on the role of an organism that can shoot!

### Target Contact Demographics

Any age group will find this game fun and challenging, this should be a good game to burn a little time on and study some great mechanics that were put into place.

### Platforms

This game will be solely released on the PC.

## What sets this project apart

This game is very colorful and strategic and can make for a great time. Warganism strives to give the player a different experience every time. When fighting against the computer it is absolutely unpredictable and can pose a great threat when trying to defeat it.

## Core gameplay MECHANICS (detailed)

### Mechanic #1 – Acceleration / Deceleration

Players will have a realistic speed and feel. An organism in Warganism does not instantly come to an instant stop. Instead an organism will slide a tiny bit as it decelerates back to a velocity of 0. This gives an element of realism and causes immersion as a player will sense and get a feel for the movement of an organism.

### Mechanic #2 – Projectiles Emitted

Players will be able to shoot at each other by using their fire buttons. This projectile will consistently move in whatever the direction the organism may be facing at that current time when the fire button is pressed. There will be a few math calculations to get the rotation of the current organism and converting that rotation into a velocity for the projectile. The same logic we use to rotate the organism will also be applied to the projectile to make it appear as if it is travelling correctly. This is a great feature that gives a lot of realism and fun elements that can be utilized at the player’s control. The projectiles will be colored the same as the organism using a programmed color filter that will apply over the white base projectile. When a player is hit by the opponent’s projectile they will shrink a bit. If a player happens to find themselves getting hit a lot they will shrink to the size of 0 and the enemy will win.

### Mechanic #3 – Pickups Make You Stronger

Players will pick up blobs as they spawn on their side of the screen. These blobs will represent the opponents color and seem like the opponent is infecting the other side of the petri dish. These pickups serve the purpose of making you grow larger and thus having larger shots to improve your chance of hitting the enemy. Players will get ammo (3) per blob they pick up and this will keep the game interesting as one will have to shoot wisely. The AI controlled organism will also pick up blobs making him/her larger and more dangerous. The CPU has a trick up its sleeve though and this will be seen as the amount of ammo drops into the negatives.

### Mechanic #4- AI programmed organism

This will be a mode the player will engage on the main screen. Once the Mode is “P vs CPU” a player will face the challenging, cheating, artificially intelligent organism. This organism will be programmed to move around the screen randomly collecting blob pickups and if they have ammo they will aim at the player and shoot. The AI will be programmed revolving around a state machine and place the CPU into a state which will generate actions. For example when in the state of “Move Random” the CPU will find a random point on his side of the screen and set it to its new “Destination” then it will be set into a state of “Moving”. After moving a bit the CPU may want to check if it has ammo and aim at the player. Some vector math is done at this point and sets the new CPU destination to the player’s center position. One mechanic the AI has is that it will only fire if it is looking in a small range window of the player. We will calculate this aiming window by creating a vector from the CPU to the player and use the CPU’s velocity vector to see if the two organisms are facing each other. When performing the Dot Product on the two normalized vectors we arrive at a number that can be used to check if the AI is aiming within a range of the player’s position.

### Mechanic #5- Player Vs Player

From the main screen a person may choose the mode to be “P vs P” which stands for player vs player and is globally known as that in the gaming culture. The 2 players will be set to a screen where player 1 (keyboard) will choose their color, then player 2 (mouse) will do the same after. The two players will now be assigned to their side of the screen or shall we say petri dish and commence battle.

### Mechanic #6- Dynamic Slime Tail

As a visually esthetic element solely, the slime tail will be unique to each organism. This will also be of the same color as your organism and trail in behind the player giving them a great sense of movement.

This feature mostly adds to the immersion and visually appealing aspects of the game. Depending on the movement of the player one can watch the hypnotic tail retrace its path back to the player. This tail will scale bigger and smaller dynamically as the player grows/shrinks in size.

## Story and gameplay

### Story (BRIEF)

- Trapped in a Petri Dish!

- Protect yourself from your enemy organisms!

### Story (Detailed)

In no way can I create an interesting story to mask what is going on in this game. I guess I could say something along the lines of … “You were a scientist working in your lab one night when you discovered an ancient secret the Egyptians use to use. When applying this secret to your work in the lab you discovered you could shift your consciousness into the life of an amoeba, thus controlling its movement. Not far after this discovery were you showing your friends and letting them control other amoebas’. This new found skill provided for great epic battles while having some drinks and partying. People started to love battling “their” amoebas and it became a fun game to play like the well-known Marble fab. Then one Day you discovered that your computer had infected the life of one amoeba and it began to epically fight back and defy the amoeba laws that were thought to be set in stone. This offered a new challenge to fight and try to beat the computer possessed amoeba. “ :P

### Gameplay

Player will either fight against one another or against the AI. A player will have to position themselves well to dodge incoming projectiles. They themselves at the same time will be shooting projectiles at their opponent. This game will have many dynamically scaling effects such as size of player and shot size. The player will move around the petri dish gaining size while trying to avoid being shrunken by an enemy projectile.

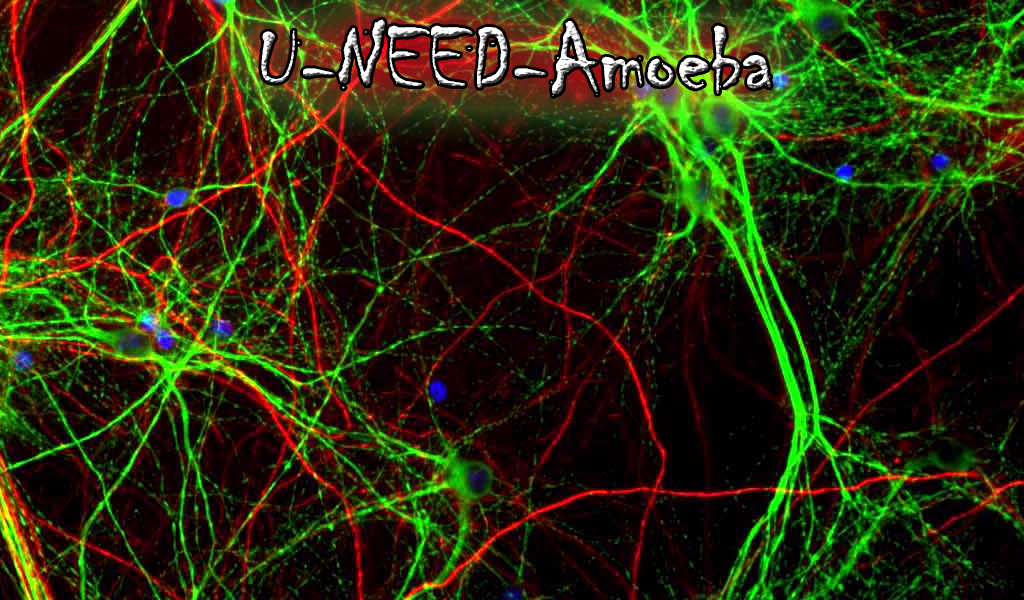
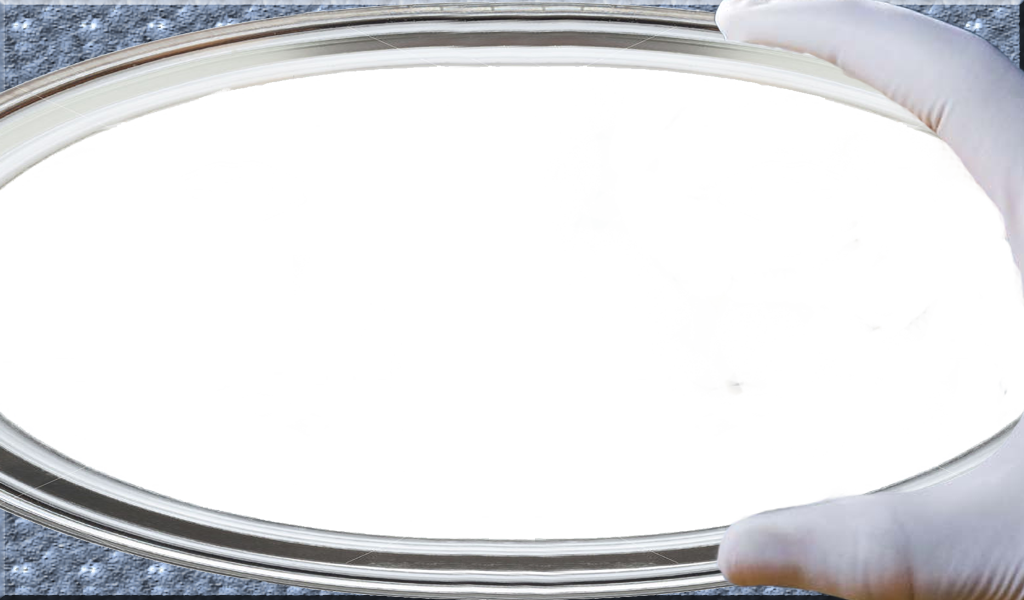
## Assets

### 2D Textures

| Resource | Role | Source |
| --- | --- | --- |
| Buttons | Interaction, Navigation | Created by Photoshop CC |
| Backgrounds/Menus | Visual appeal / Emersion | Created by Photoshop CC |
| Sprites | Character Animation/ Effects | Created by Photoshop CC |

### Art Samples

Main Menu Screen Amoeba (Players) Pick-Ups

Buttons Character Screen Background Game Screen Background

### SOUND (Ambient)

There will be main game music with a funk style beat to it.

### SOUND (PLAYER/COLLISION

When a projectile hits a player, it will produce a splat sound. When a player grows it will have a grow sound. When a player shoots a whoosh sound is played.

### CODE

Button class will be a class which allows creation of buttons. The game entity class will be the class which creates all the in game entities. Projectiles, pick-ups, slime, players, AI and buttons will all be in game entities. The class will hold all the functions needed to render, scale, update, animate, move, apply button effects, rotate, handle collision and handle all the AI decision making and handle all the setting and getting of properties for the game entities. A math utility class will be added as well, which was provided to us by Rob Cote. A Window class which will handle the creation of the window that will render the game. The particle class will handle the creation of the slime trail that will be on the players and CPU. The particle class will house a function that fades the slime out over time.

The pick-up class will handle the creation of pick-ups and won’t have any functions just properties. The projectile class will handle just creation and have no functions as well. Only properties. A random number generator class will be included as well. This was provided by Ross Driedger.

A SDLHelper class will be created to handle all the core game functionality. It will handle the switching of game states, handle mouse and key presses, updating of the game and game entities, showing of screens based on which state the game is in, spawning and rendering of all the game entities.

There will be a state machine class which will house a struct of pointers to enums of states. They will be global and can be accessed by any class.

### ANIMATION (Environment)

Not much in terms of environment animation. The buttons will have a grow and shrink effect when you hover over them.

### ANIMATION (Character)

#### Player

Player will have a slime trail that will have its alpha reduced over time. Character projectiles will squish on the x axis to give the illusion that the enemy being hit is absorbing the shot. Player will grow from picking up the pick-ups that spawn and will shrink after being hit.

#### NPC

Same as player.

### Menu and Game Layouts

Main menu will consist of 4 buttons. The play button which will take the player to the character selection screen. A music toggle button, which will turn the music on or off. A sound toggle button which will turn sound fx on or off and a game mode selection button, which will toggle between Player versus Player and Player versus CPU.

Character selection screen will consist of 17 buttons that each have a different colored amoeba for the player to pick from.

Game screen has a playing field which is divided in half and on the left side will house player 1. The right side will house player 2. Current ammo will be displayed at the bottom of each player’s playing field. The current size will be displayed at the top.

Pause screen will have 2 buttons. One will allow the player to restart. The other will allow the player to quit. Pause screen will also have the word “PAUSED” in the middle of the screen.

Player 1 win screen will be the same as the pause screen, with the exception of the word displayed. Instead it will display “Player 1 Wins”.

Player 2 win screen will be the same as player 1 screen, except it will display “Player 2 wins”.

CPU win screen will be same as the players win screen, except it displays “CPU Wins”.