|  |
| --- |
| Prototyping Report |
| Group 2 |
|  |

# Prototype: Gameplay

The core gameplay is fundamental to a good game. This determines for a large amount how much fun the player has in playing the game. This prototype was used to develop gameplay and specifically the gravity shift system.

### Does it work as expected?

Based on playtesting, a lot of changes have been made to the gameplay. This was not because it didn’t work as expected, but because the changes improvement the gameplay experience. We had a big problem with the gravity switch; the player lost its orientation if the gravity changed. This was solved, and circumvented by rewriting the code with ray tracing smooth lerp transitions. Camera and movement are based on basic FPS movement. Whe are still experimenting with the function of the different mouse buttons. We decided to split the function of the bullet to one bullet that can hit other players (Left mouse button) and another bullet to make the gravity switch. (Right mouse button)

There is still a bug with movement. (when you stop moving you go back a little). This is a minor annoyance we where not able to fix yet.

### Did it take more time to develop than expected?

Gameplay is something we expect to work on for the rest of the development. We did not experience major set backs.

### Are we satisfied with the prototype?

The game is already fun to play and move around in. This is very positive and we are satisfied with the result we achieved in the first two weeks.

### Will we use it in our final game?

This is the starting point we will base our game upon. All gameplay elements are subject to change, but we do not expect to alter a lot.

### Does it need improvement, and why?

The multiplayer aspect needs to be merged to do further testing and adjustments. The core movement is fun to play and does not need an improvement as of now.

# Prototype: Procedural Level generator

The initial idea was to program a procedural level generator that was not grid based. It would pick a given number of initial spawn places (the user could input how many) and it would spawn a cube there. The cube itself would have a script attached that chooses one of the cube’s neighboring positions and spawns another block there (which also has the same script attached, so it’s like a recursive algorithm). It would send a parameter to the next block indicating how many blocks should still spawn (could also be adjusted by the user).

### Does it work as expected?

The concept worked, it would create levels, but there were quite a few issues.

* It was difficult to prevent the algorithm from growing outside the defined dimensions of the level.
* If 2 spawns grow towards each other, because it isn’t grid based, the overlap between spawns may be rather awkward, which would not favor gameplay
* There was a bug where if I tried to implement a “favored direction” into the algorithm (e.g. the spawns are more likely to grow in x-direction) it wouldn’t quite work.
* There was a bug where sometimes blocks would be spawned multiple times at the same location.

As we tried to fix these bugs, we realized that it would probably just be better to rewrite the algorithm and use a grid based algorithm that fills up a matrix with 1’s (representing blocks), and applying a separated draw function that instantiates blocks at the positions indicated in the matrix. This would certainly solve the problem of blocks being spawned multiple times at the same location (the draw function would only go over all matrix elements once, and thus only draw a block at most once in a certain position). This would also make it easier to force the algorithm to stay within the indicated dimensions of the level, because you simply can’t define 1’s outside of the matrix space. We also discovered the bug causing the preferred direction algorithm not to work properly and fixed it. Additionally, the new algorithm is a lot quicker than the old one. You don’t want people to wait for a minute before the level is created and they can play. The old algorithm used colliders to see if the next position was free to spawn a block, this takes a lot of time. It is much quicker to just check whether an entry in the matrix is 1 or 0.

### Did it take more time to develop than expected?

Definitely, especially because we had to rewrite the entire algorithm after realizing the first idea was not a very good idea. Also the amount of bugs was larger than expected.

### Are we satisfied with the prototype?

Yes. The levels created are really creative and work well (has been tested in the actual game).

### Will we use it in our final game?

Certainly. Procedural level generation will prevent people from becoming to familiar with the levels and always picking spots that work well, you have to always get used to a new level, and in this sense everyone is equal in knowledge about the level.

### Does it need improvement, and why?

As of right now there are no things that need to be improved in the algorithm.

# Prototype: Multiplayer

Surreal Gravity is intended as a multiplayer game; single player will most likely be an option, but multiplayer will be the core aspect. Therefore this prototype has been made.

This prototype consists of 2 parts: Networking and Accounts/Account storage. The networking part was finished first; the accounts part second.

### Does it work as expected?

*Networking*(Note: as of now, this prototype has been tested on 1 PC (thus not tested if 2 PC’s can connect to each other)   
One game client has the option to either create a server or connect to a server (via the same network, so LAN). If a server has been created, another game client cannot create a server (since the server names are the same). A game client can refresh hosts; if a server has been created, the game client can connect to the server. You can only control 1 ‘player’ on a game client.

It works as expected: if the player moves on one game client, every other game client sees the player move. If one player (doesn’t matter which one) comes close to the square, the square turns blue on every game client. There is thus obviously a connection, and via that connection all players can simultaneously move in the game.

*Accounts*If a server is created, or a connection with a server has been made, you have to login to an account. If needed, you can first register one, and then log in on that account (or another one).

The accounts are as of now stored locally in a text file (simply username \n password \n username \n password and so on). Our goal is to store the accounts on a webserver (preferably in a database), but right now not much is clear about the webserver which should be provided.

If you are logged in, your player spawns and your username is shown on the top of the screen. The color of your username depends on the team you’ve chosen. If a game client connects, and logs in, every other connected game client sees the player connecting through both the spawning of the player as well as the manifestation of his username.

It works as expected: you can login to an account (given that the account exists and the password is correct) and everyone knows which player you are due to the networking.

### Did it take more time to develop than expected?

This prototype took more time to develop than initially was thought. The networking part was finished quite fast, however the accounts part took more time to finish. More specifically, being able to show your username to every other client was more complex than we had thought. Due to this, it took more time to finish the prototype.

### Are we satisfied with the prototype?

We are quite satisfied with this prototype. We should be able to implement this prototype in our final game, with only a few adjustments. The core of multiplayer is already there; you can connect to each other and you can log in to accounts.

### Will we use it in our final game?

We will definitely use this prototype in our final game; as previously mentioned, multiplayer is the core aspect of our game.

### Does it need improvement, and why?

This prototype needs some improvements.

In this prototype, the accounts are stored locally. This is not preferable when playing via LAN (if 1 client registers an account it will only be stored on that client’s pc). We want to store the accounts on a webserver, if possible in a database. We will probably also look into encoding the passwords, to prevent people “hacking” our game.

In this prototype, you have to connect first, then log in and then you are spawned. This is not ideal; hence, we will probably implement a sort of lobby; here you can connect with other players while already logged in to an account. This gives the possibility of playing single player locally, which is important. This also makes sure that the server initializes everything before the game is loaded and everyone joins. Moreover, you can, for example, first choose your character before playing. And lastly, it will make sure the game start when everybody is ready (and not that the server is playing already when he is alone).

# Prototype: Models

The current prototypes of the models in our game are:

* Bullet
* Character
* Weapon
* Flag

### Does it work as expected?

At first the futuristic bullet was created using a puck as core; however we decided that it would look better if the core was a ball. The bullet was also animated with easing on and had to be turned off. The animation of the character and bullet are now working as expected. For the weapon we created a RPG at first, after some reviewing a futuristic weapon would be more fitting in our game. During the making of the weapon we encountered a strange problem: some parts of the weapon suddenly disappeared when joining the parts. To fix this problem a few redesigns had to be done, now the weapon works as expected. The flag is a pretty simple blender model, so that worked pretty well. The problem with it was with Unity´s built-in interactive cloth. Since the Mesh provided by Unity is a plane, the face of the flag is only seen from one side. For now we have fixed this by making two of these faces with the backs against each other. This is still not a perfect solution so we´ll definitely look into changing this later on.

### Did it take more time to develop than expected?

Because of the problems we encountered it took some more time to develop than expected.

### Are we satisfied with the prototype?

At this point we are satisfied with the models.

### Will we use it in our final game?

Yes, these prototypes will be used in our final game.

### Does it need improvement, and why?

The models definitely need some improvements and optimizations, a texture and mapping have still to be added to the character.