1. # Surreal Gravity, Group 2

2. The theme is procedurally generated levels. We want the playing experience to be different each time, and therefore the levels will be created with a procedural algorithm.

3. The idea for our game is a first person shooter. The level will consist of blocks (cubes) that the player can walk on. The direction of gravity can be controlled with your gun. If you shoot on a face of a cube, the inward normal of that face will be the next direction of gravity. This means that in total there will be 6 directions of gravity, 2 directions in x, 2 directions in y and 2 directions in z direction gravity (which of the 2 directions in a certain axis will be chosen is determined by the inward normal of the face of a cube that is shot). The change of gravity will only be applied to the shooter, which means every player will has its own direction of gravity. We might add different game modes, such as a race version where the players have to race and win by finishing first, or an actual shooter where players need to kill each other and win by being the only one left. The idea is to limit the amount of bullets a player can shoot, for example by using a timer or not allowing a player to shoot a new bullet whilst the old bullet hasn't touched anything yet (and of course, use a timer on the bullet that destroys it after some time to prevent a player from never having a bullet again if the bullet missed everything).

4. # Components

**Computer graphics**

- 3D models. All of the models will be made in Blender. The models are for example: playable characters, weapons or bullets. They will also be provided with textures ★ (Kwok)
- 3D animated models. Animations will be made of the character. The bullet will be more like a sci-fi futuristic bullet and will be animated as such. ★★( Kwok)
- Sound effects. These will improve the ambience while playing the game. They will exists out of single sample sounds (firing a bullet, killing someone, etc) as well as the background music which will be selfmade. ★★ (Ayyoeb)
- Particle systems. Special effects such as disintegrating blocks will be generated in Unity, blocks will disappear throughout the game to add a more dynamic gameplay. ★(Coen)
- (Playing with) Lights and shadows. We will try to play with the lights and shadows to create a good atmosphere. For example the bullets will be light sources as well. ★ (Coen)
- Start, pause, end screen. An appealing menu screen will be created in Photoshop. ★ (Ayyoeb)
- High scores. Keeping track of the high scores as well as the actual scores while playing the game). The high scores will be accessible via the menu. ★ (Coen)
- Options. For example: enabling/disabling the sound, adjusting the brightness.★ (Ayyoeb)
- Instructions. The goal of the game (being the last one alive) and the keys that are used will be explained here ★(Coen)
- Credits. All the team members with their functions will be displayed.★(Ayyoeb)

Total: 12x★

**Artificial Intelligence**

- Genetic algorithms, for example to train the procedural generation algorithm to make more viable/fun levels. The idea is to have a certain amount of variables, like how many

blocks will be present in the level, how big the dimensions of the level are, how sparsely the blocks are distributed in the level, and build levels with these variables set to random numbers in a suitable range. We will playtest these levels and manually give them a quality score. Possibly we might speed up this process by playing only very shortly and giving quality scores based on first impressions. Use this to train an algorithm to find good combinations of these variables to get good playable levels. This can be a lengthy process, hence 4 stars, possibly even 5. ★★★★(★) (Steven)

- Some AI algorithm to train an artificial enemy. We don't know yet how the AI is going to move about yet, or whether it is going to be moving at all, so this is just an idea. If this proves too difficult we might still have to hard code it. The idea is for example to have a sort of "capture the base" kind of game mode that can also be played in single player mode. In single player mode we might then have towers standing on one of the forts that will shoot at you to defend the fort. We can perhaps train the shooting with one of the artificial intelligence techniques (like Hebbian Neural Network). Again this is like a small project on its own, so 4 or 5 stars. ★★★★(★ ) (Roberto)

**Total: 8-10x★**

**Web & Databases**

- Collect playthrough data. For example we can retrieve from the server who won and with how many points or how fast etc. Perhaps we can add usernames where each username has individual scores. Maybe we can also add indications of how skillful a certain player is (e.g. Kill/Death ratio, Win/lose ratio). This should be a moderate amount of work, hence 2 stars. ★★ (Roberto)
- Store data on web server. For example we can store on the server who won and with how many points or how fast etc. Again we could work with usernames. ★★ (Roberto)
- Visualize data on web server. Find a GUI representation of the stored data. ★★ (Kwok)
- Collect and show highscores from web server. ★★ (Kees)
- Online gamer accounts with avatars. It should be possible either to add personal avatars or to choose from a set of given avatars. Or perhaps even make it possible for users to interact with a certain standard image to customize it into their avatar. Depending on which implementation we choose this might require more or less work. An approximation of the work is 3 stars. ★★★ (Kwok)

Total: 11x★

**Programming**

- Procedurally generated levels. Each level should be different and therefore procedurally generated. This should be bound to some restrictions of course, to prevent levels becoming unplayable. Depending on how advanced the procedural algorithm is going to be this could be anything from 2-5 stars. An average of 3 stars is chosen. ★★★ (Steven)
- Moving platforms. For example some of the cubes in the level may disappear after some time or some of the cubes may move in space. Given the fact that the level is procedurally generated this might pose some difficulties, therefore 2 stars. ★★ (Steven)
- Multiplayer. Probably LAN multiplayer via a local server. This can be a very lengthy process, hence 5 stars. ★★★★★ (Kees)
- FPS independent. The game mechanics should always be equal regardless of how fast the computer is. ★ (Roberto)
- Game variables can be changed by player. For example the amount of effect gravity has (a lot of gravity can mean difficult to control), or the maximum speed you can achieve in space. The point of this is to allow a player to alter the feel of the game. High gravity and maximum speed can make the gameplay difficult, but also can make the game more fun and intense to play. This should be done in for example an option menu. ★ (Kees)

- Physics. Several things. Artificial gravity that changes direction. Collision detection between bullets, the level, and players. Movement of the player, etc. ★★ (Roberto)

Total: 14x★

5. Steven van der Helm, S.vanderHelm@student.tudelft.nl, Producer
   Roberto Mol, R.G.Mol@student.tudelft.nl, Lead Programmer
   Kees Kroep, k.kroep1@gmail.com, Game Designer
   Coen Berssenbrugge, C.W.J.Berssenbrugge@student.tudelft.nl, World builder
   Ayyoeb Ichaarine a.ichaarine@gmail.com, Lead Audio Design
   Kwok Hou Man kwokhou.nl@gmail.com, Lead Artist

6. **10 Nov – 16 Nov:**
   - Create the algorithm for procedurally generated levels.
   - Have a basic first person view that can be controlled.
   - Start creating 3D models.
   - ***Deliverable: Core Project Document***

   **17 Nov – 23 Nov:**
   - Finish and test all objectives so far.
   - Start creating sound effects.
   - Start working on animations for the 3D models.
   - Adapt the procedural algorithm to include levels with animation/movement.
   - Start working on allowing user to create a custom avatar.

   **24 Nov – 30 Nov:**
   - Finish and test all objectives so far.
   - Enhance 3D models further. Start adding textures.
   - Start setting up server, how to add and obtain information from this server.
   - Start working on allowing user to give custom texture to use on his/her character.
   - Start working on the computational intelligence techniques.
   - Revise vision for our game using newly obtained insights.
   - Start thinking about different game modes.
   - Write the Game Design Document.
   - ***Deliverables: Game Design Document***
   -

   **01 Dec – 07 Dec:**
   - Finalize interaction with server, collecting, storing and using data from and to the server.
   - Allow user to create accounts and link the custom avatars to the account.
   - Start putting things together. Start adding models, sounds animations, access to server etc. into one game. Make everything work together smoothly.
   - Start thinking about ideas for applying computational intelligence algorithms to artificial enemies and possibly to the procedural generation of the level.
   - Start working on implementing different game modes.

   **08 Dec – 14 Dec:**
   - Continue building the game.
   - Start adding option menu to the game that allows changing some settings of the game.
   - Continue working on the computational intelligence techniques.
   - Finish working server and allow access to server.
   - Continue working on different game modes.
   - Work on allowing LAN multiplayer.
   - Fill in the peer reviews.

- ***Deliverables: peer reviews***

**15 Dec – 21 Dec:**
- Continue working on different game modes.
- Work on allowing LAN multiplayer.
- Continue working on the computational intelligence techniques.
- ***Deliverables: early access game***

**05 Jan – 11 Jan:**
- Finish the computational intelligence techniques and adopt them into the game.
- Finish the different game modes and have them working.
- Finish LAN multiplayer.

**12 Jan – 18 Jan:**
- Put everything built so far together.
- Start performing final tests.
- ***Deliverables: beta game***

**19 Jan – 25 Jan:**
- Fix final bugs, tweak the game for optimal play.
- Final testing of game.
- Prepare presentation of game.
- ***Deliverables: peer reviews, and the final game***

7. **Link to github:**
   https://github.com/kkroep/Surreal_Gravity.git