Changes indicated in red

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. There will be 2 different game modes that the server can choose between. The modes differ in the weapon that all players have. One of the weapons will be a railgun, with which you can instantly shoot other players from a distance (similar to the gravity change gun). The other will be a fork, which is a melee weapon, meaning players can only hit each other from short distance (this will require the players to use the gravity more and chase each other). The fork can only hit players forward so that gravity changes really need to be used. We also included little robots that fly around the level and destroy the procedurally generated blocks and by doing so alter the level.

4. # Components

Computer graphics

- 3D models. All of the models will be made in Blender. The models are for example: playable characters (normal and gingerbread), the robots, weapons or ~~bullets~~. They will also be provided with textures ★★
- 3D animated models. Animations will be made for the character. The bullet will be more like a sci-fi futuristic bullet and will be animated as such, this is now used to decorate the railgun. ★★
- Sound effects. These will improve the ambience while playing the game. They will exist out of single sample sounds. They are either handmade or downloaded with free license and then combined and modified. (~~firing a bullet, killing someone, etc) as well as the background music which will be self-made.~~ ★★
- Particle systems. Special effects such as a smoke effect for disintegrating blocks will be generated in Unity, blocks will disappear throughout the game to add a more dynamic gameplay and robots will have trailing particles too. ★
- ~~(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. ★~~
- ~~Start, pause, end screen~~. Main Menu. An appealing menu screen will be created in Photoshop. We make use of various input fields and use our own button effects in the new UI. ★
- ~~High scores~~. Score screen. Keeping track of the high scores as well as the actual scores while playing the game. Also deaths, kills and player names are visible. The player can view the screen by pressing tab or when he dies. When the player dies he can see the killer on the score screen while waiting for a respawn. ~~The high scores will be accessible via the menu.~~ ★★
- Options. For example: enabling/disabling the sound, adjusting the ~~brightness~~ mouse sensitivity.★

- Instructions. The goal of the game (being the last one alive) and the keys that are used will be explained here ★
- Credits. All the team members with their functions will be displayed.★
- To make the level less monotone, we have created a bunch of different textures that are tillable when put next to each other. A random texture of this set is selected when the blocks are instantiated. The textures are hand-drawn and still tillable. There are four different texture themes, with their distinctive look and their own tillable texture set. ★★★
- The robot features hand-drawn animated textures when drilling ★
- Music in game: The music in the game is self-made. The music in the menu will prepare the player for the game. The music while playing will boost the adrenaline of the user and makes the game more exciting to play. There are up-tempo parts in the tracks for variation. We iterated a lot on the music to get to the current level of quality ★★★

Total: 20x★

Artificial Intelligence
- For the robots that fly around we need animations that are queued at the correct time. So the robot needs to determine when the animation is queued. ★★
- We need a selection mechanism that selects blocks in the level that the robot needs to fly to and destroy. These blocks should not be random completely because we don't want to decrease the quality of the level. So the robot will only pick blocks that are not part of continuous paths, such as not to break these paths. ★★
- We need a path finding algorithm that needs to find a path from the robot's current position to the selected block. This is a lot of work, hence 4-5 stars. ★★★★(★)

Total: 8-9x★

Web & Databases
- Allowing players to register accounts on the webserver and also allowing them to log in via the web server with their personal account. ★★
- Collect play-through data. We will retrieve all the victories of a certain player, the amount of games played and finally the win-lose ratio. ~~and we might collect Kill-Death ratios for the played games.~~ ★★
- Store data on web server. ~~We will log for every user the total amount of games played, the wins of the player and possibly average Kill-Death ratio of the player.~~ We will also log all games played, which player wins in those games and keep track of which players participate in which games. The wins and losses of individual players can be reconstructed from this information using SQL views. ★★
- Visualize data on web server. Not a GUI representation, so only 1 star. ★
- Collect and show high scores from web server. We choose to implement the GUI representation in the game itself. Providing useful and competitive information for the active account ★★

Total: 9x★

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. ★★★
- Multiplayer Synchronization. ~~Probably LAN multiplayer via a local server~~. ~~This can be a very lengthy process~~. Every single thing that has to happen on all computers in the same way needs to be programmed for multiplayer. This is without a doubt the most

challenging and time consuming task in our game. (e.g. animations, scores, death, procedural levels, custom textures, robot movement, etc.) hence 5 stars. ★★★★★
- Multiplayer Connection. The players need to set up a host without difficulty, and clients need to find the host and connect to it. This is very complex and took a long time to get right. Hence 2 stars. ★★
- FPS independent. The game mechanics should always be equal regardless of how fast the computer is. ★
- ~~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 gameplay more fun and intense to play. You can also for example edit the amount of robots that destroy game blocks in the level. This should be done in an option menu. ★ (Kees)~~
- Physics. Several things. Artificial gravity that changes direction. ~~Collision detection between bullets, the level, and players~~. Ray-tracing for the gravity changes. Movement of the player, not trivial jumping, custom physics materials, etc. ★★★
- Different game modes. One with a railgun and one with a fork (melee weapon). This is a completely different gameplay. It required completely new mechanics to be programmed for the player. Also the other game mode has been polished for smooth play. Both gamemodes also feature different character and weapon design ★★

Total: 16x★

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.
   - (Not yet done, will be later concern) Start working on allowing user to create a custom avatar.

   24 Nov – 30 Nov:
   - Finish and test all objectives so far.
   - Already start working on putting components together.
   - 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 selection algorithm for the robots.
   - 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.
- Integrate access to the server into the game.
- (Already done) Start thinking about ideas for applying computational intelligence algorithms to artificial enemies and possibly to the procedural generation of the level.
- Start working on the path finding algorithm for the robot.
- Start working on implementing different game modes.

08 Dec – 14 Dec:
- Continue building the game.
- Start adding animations to the robot's movement.
- 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