Core Project Document V1.3

Group 6 Escape

November 2014

1 Group Roles

Game Designer	Xander de Ronde	4170334	X.E.J.deRonde@student.tudelft.nl
Lead Programmer	Tim de Boer	4007352	T.deBoer-2@student.tudelft.nl
Lead Artist	Jordi Hessels	4155343	J.F.B.Hessels@student.tudelft.nl
World Builder	Arnoud Jonker	4170431	A.Y.A.Jonker@student.tudelft.nl
Producer	Mike Koch	4232313	M.R.Koch@student.tudelft.nl

2 Theme and interpretation

You only get one chance to live, don't spend it locked away. The core goal of the game is escaping from some sort of high-security prison and breathe fresh air again. The main character is serving a life sentence, but due to sudden circumstances an opportunity to break out presents itself. This is likely his only chance to ever return to the outside world again.

3 Game idea

The basic idea of the game is to escape from a building, that consists of multiple rooms, each with their own function. The game will feature an art style that is inspired by animation movies. The gameplay consists of escaping from a high-security prison, in which the player should stay undetected as much as possible, to ease their escape. This can be done by sneaking around, but the player will also find several objects in the environment that could aid his escape. This results in gameplay that will combine precise movement in the shadows with finding an optimal way to stay entirely undetected or find a quick way out. The room layout will be procedurally generated, meaning every playthrough will play out differently to keep the experience entertaining.

4 Key components

Computer Graphics(16 stars):

- 3D models
    1. 3D-models (*) The game will contain 3D-models for most objects, such as furniture and objects which could be interacted with. - Jordi
    2. Animated 3D-models (***) Some 3D-models will be animated to make the game feel more alive. The player and most enemies (both robots and live creatures) will have animations. To satisfy the demands, a lot of animated models will need to be created. - Jordi
- Texture
    1. Procedurally generated textures (****) By generating textures through algorithms, repetitive looking textures can be prevented. This will be challenging to pull off in a way where the texture quality itself isn't sacrificed all too much. - Arnoud

- Special Effects & Juiciness
    1. Sound effects (*) Some freely available sound effects could be used in the game, but we intend to also record some sound effects ourselves. - Jordi
    2. Particle Systems (*) To make the game look better aesthetically and bring the attention of the player to the right elements, particles could prove very useful. - Jordi
- Rendering
    1. Play with lights and shadows (**) The player may be able to hide in the shadows, while light may help for overview of the room. Light can also be used to emphasize important objects in the room. - Arnoud
- User Interface (with new Unity3D UI tools)
    1. Start, pause, end screen (*) It is almost impossible to create a user-friendly modern game without these elements. - Xander
    2. High scores (*) The high scores are saved from the best performing players. Playing through the game faster, finding more creative solutions or staying undetected the entire time will result in a higher score, whereas getting detected could deduct points. - Xander
    3. Options (*) The options menu will contain elements like adjusting volume and sound, but also advanced graphical options like Anti-Aliasing or resolution. - Xander
    4. Credits (*) Completing the game leads to a screen where the group members or outside sources will be credited for their contributions. - Xander

Artificial Intelligence(6 stars):

1. Pathfinding using our own method (***) We will create our own pathfinding algorithm, to make enemies search for the player or patrol around an area. By using this, guards will surely feel much more lively, when encountered. - Arnoud
2. Enemy you always lose from (***) A possibility for using an enemy that will always defeat (or in this case, detect) you could be a grand reward, guarded by an 'unbeatable' guard. To reap these rewards, the player will have to find a way to outsmart the guard. If the player doesn't, he will almost instantly be detected and the guard will sound the alarm, increasing the difficulty significantly. The challenge in making this mechanic work in a way that doesn't frustrate the player lies in balancing the ways to defeat this enemy: if it's too easy, the game as a whole won't feel challenging, but if the encounter is too difficult, the game could work infuriating. - Arnoud

Web & Databases(17 stars):

1. Collect playthrough data (**) During the game, some interesting data of the playthrough will be collected. One could think about scores, average game times, collectables, the amount of times the player was detected, but also more technical data like average framerate (or deviation from that average in certain areas of the level). - Xander
2. Store data on web server (**) All collected data must be saved on a central web server with a database. - Tim
3. Visualize data on webserver (**) All collected data could be visualized. For example trend-lines and graphs could be generated to show where players are encountering many difficulties, or certain elements being too easy. - Mike
4. Collect and show highscores from web server (**) The game must be able to retrieve the current high scores and present them in a highscores menu. To extend upon this feature, the highscores could also be presented on the webpage, for live statistics. - Tim

5. Online gamer accounts with avatars (***) Based on unique identifiers, like a username and password, the player could have his statistics saved online. In game a player could chose default avatars and/or upload custom avatars. This would be handled through the website and shouldn't be too much work on top of what would already be in place. - Tim

6. Save and share game states with others through social media (***) A player could share his highscores or other statistics on social media. This could allow players who are interested in this specific feature to find a way to compete over social media. Additionaly, the server could "share" and "tweet" daily the top 3 gamers.

7. Mobile device as second screen (***) A mobile device could be used to show the current inventory, i.e. the items currently in the player's posession, and to give explanation about the objects in your inventory. These pieces of information would also be available in-game, but using a second screen could prove to be a nice feature for some players. This feature would only extend upon features that are already in place, so it could be partially 'recycled' - Tim

Programming:

- Game Mechanics(13 stars)

    1. Procedurally generated levels (***) The layout of the levels will be procedurally generated, i.e. the way in which certain types of rooms (security room, hallway, armory etc.) are connected. The rooms themselves would also be generated procedurally, where there is a set of possible objects in the room, and they could be placed differently ever time. The challenge here lies in making the levels feel fresh every time, while also maintaining a comparable gameplay experience. - Arnoud
    2. Race against clock (*) An additional gamemode could be a race against the clock, where the play could be forced to complete a level in a certain time. This could steer the player into some quick decisions, which could or could not work out, contributing to the excitement factor. - Xander
    3. Online multiplayer (*****) We want to implement a multiplayer mode where the players have to work together to escape the building and win the game. Adding this would make the game much more enjoyable, as it proves a different kind of challenge: not only will the possible solutions be more complex, they would also require some co-ordination between both players. Because this feature will require some more complex puzzles/challenges as well as more complex gameplay mechanics, this feature will take a lot of effort. - Tim

- Game Loop

    1. FPS independent (use Time.deltaTime) (**) The game time will be fps independent, so the game doesn't behave differently based on a machine's performance. - Jordi

- Physics

    1. Unity's triggers only for collision checks (*) Using Unity's triggers for collision checks allows for clear behaviour in when a certain event is triggered. This could be used to for instance pick up objects. - Jordi
    2. Use Unity's full physics simulation for all movement, collisions etc (*) For walking around solid objects and giving certain collisions specific effects, Unity's physics simulation will be used. - Jordi

5 A rough schedule

Week 2.1:	Initial exploration of possible game concepts and further development of said game concepts. After that, initial prototyping of game play elements will start.
Week 2.2:	Further prototyping of game play elements and game mechanics.
Week 2.3:	Further prototyping and finishing prototyping, adjusting Core Project Document.
Week 2.4:	Start development of the game, using the prototypes created before. Creating a working prototype of the actual game.
Week 2.5:	Further development of the game, implementing new ideas into the base game if any were found and expanded on.
Week 2.6:	Further development, updating documents accordingly. Early access version will have to be finished by now.
Week 2.7:	Further development.
Week 2.8:	Further development, game entering Beta stage.
Week 2.9:	Further development and release of the game.

6 Link to the GitHub project page

https://github.com/tim427/minorproject/