|  |  |
| --- | --- |
| **Marking Criteria** | **Describe how your game matches the criteria (Description of each item is limited to 50 words)** |
| **Game theme and design (5%)** | |
| Game story, and your choices of videos for developing the story: | The game is based on the Assassin Kitten video in which the goal is to assassin the targets with the kittens who are dropped into the scene by an Eagle.  The first level of the game is based on the Boris got stuck video which simply involves assassinating Boris Johnson who is stuck on a zipwire  The second level is based on the Peppa Pig in London video in which the Queen drives around a tour bus full of children. Again, the goal is to assassinate the Queen however this time you must avoid killing the children as well. |
| Choice of game type and how it matches with your game story: | It’s a 2D physics game similar to something like Angry Birds. This works well with set levels and allows a lot of flexibility in item design as every item is simply a physics object in the game. The levels can also easily be designed only using a few set items which makes level design much easier whilst still leaving plenty of room for variation and imagination. |
| **Core development and implementation (30%)** | |
| Game scene (visual representation [2D, 2.5D or 3D], internal data structure): | Every scene is 2D with a fixed camera focusing on the level. The level design is simply stored in a grid of blocks withing the Unity Scene File. Each scene will always contain the Eagle which drops in the Cats, a ground plane, a background image, and a basic UI to control the game. |
| Game flow and how it is designed (e.g., scene navigation, level design): | The scene navigation is control through a simple menu UI. Each scene is a playable level in which the required targets need to be killed in order to win the level. If the player runs out of Cats or kills a Protected Target, then the level is failed, and they can either retry or exit. Once the level is completed, the player can then proceed to the next level. |
| Game interaction (e.g., action detection and response generation): | The game is controlled simply by moving the Eagle. The Eagle drops in the Cats from above the level but can throw the them if they are released when it has a velocity. This is done by simply dragging the mouse or could be done by dragging a finger if it was on a mobile device. Once the Cat has been deployed, it then drops onto the scene and interacts with the level via a physics simulation. Items in the scene can also be damaged if they are hit hard enough. |
| Game object (e.g., use of sprite, 3D objects, object movement and animation): | Every object in the scene is a Sprite which also contains a Rigid-Body and a Collider this allows it to be seen and to interact with the other objects via the physics simulation. In some cases, blocks are animated through simply switching the sprite with another image. The Eagle is animated by quickly changing sprites to make it’s wings appear to flag and other blocks in the scene can gain damage by swapping to a different image when they get hit. |
| **Game mechanics with machinations diagrams (30%)** | |
| Main game rules / logics to control game progression, difficulty, and end game conditions: | <https://my.machinations.io/d/Accattins/02d65a408fc13d9e39ee0d7b11a11c9cb>  The game rules are very simple. The level is won when you take out the target. The level is failed if you either run out of cats or kill a protected target. Once a level is win without failed it, the player may proceed to the next level. Each level has increasing difficulty from the previous one in order to keep challenging the player. |
| Control and growth of game object abilities: | The game doesn’t actually change in terms of abilities or rules as the levels progress. Instead, the player is required to learn new mechanics with the same controls e.g., leaning to throw the cats instead of just dropping them from above. |
| **Good use of game engine (12%)** | |
| Justification of the choice of game engine (pyGame, Unity [with version number]) in terms of game theme matching and expected target audience (game player): | The game was created using the Unity Game Engine (2020.3.26f1). This is to make use of its visual interface and built-in multimedia handling tools. This greatly speeds up development and level design. I also make use of the built-in physics system via rigid-bodies and colliders which is a fundamental part of the game. The engine also provides many build options for when the game is finished. This allows me to easily export the game to multiple platforms e.g., a computer, phone, or web browser with minimal extra effort. |
| User input supported (keyboard, mouse, joystick, etc.): | The user input is controlled using a mouse with can also be simulated using touch input if it were on a mobile device. The game functions identically in either scenario. |
| Game object interaction supported (e.g., event triggering, collision detection): | Objects interact with each other via the physics simulation which affects their position and rotation. I’ve also added a collision detection system which allows me to damage and destroy blocks when they receive a hard enough hit. The game ends when targets are destroyed or if I detect that all the game-objects have stopped moving once all the Cats have been deployed. |
| Extra game engine features used (e.g., asset, incorporation of external libraries): | I’ve used several assets from Kenney Assets (<https://kenney.nl/assets>). These are all under the CC0 licence which allows me to use them in any scenario with no credit required. All of the assets used are very high quality which gives the game a professional look whilst also saving me lots of time allowing me to spend more time on the actual game design.  The game also using many of the Unity built-in packages like the Physics Engine, UI System and Scene Manager. |
| **Good use of multimedia technologies (15%)** | |
| Effective use of different types of multimedia contents and their optimization: | The game uses numerous 2D images and audio files. All of these are imported directly into Unity and compressed using an appropriate method within the Unity interface.  All the images are stored in a PNG format which is ideal for games as they are light-weight portable and handle solid colours very well which describes every single image.  All the audio files are stored as OGG files for similar reasons to the images. They are all compressed using Vorbis which helps reduce the size they occupy whilst maintaining sound quality. |
| Advanced object-object and user-object interaction (e.g., game physics, object tracking, steering behavior, multi-modal user interaction): | Every object in the scene interacts using Unity’s physics engine.  When deploying the Cats, the Eagle is tracked onto the player’s mouse so that they can precisely control the deployed cat’s position and velocity. The game camera must be used to transform the screen coordinates of the mouse into world coordinates in the game.  In order to simulate a rope in the game, I make use of Unity’s Hing Joint, the allows me to chain together rope segments in a realistic manner that makes them hang and swing. I can also utilize the Break Force parameter to cause the rope to be cut when a block is dropped onto it. |

**\*Note:** Your work must be done by yourself and comply with the university rules about plagiarism and collusion. (https://www.dur.ac.uk/learningandteaching.handbook/6/2/4/)