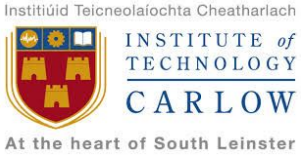
NYA(not you actually)

B.Sc Computer Games Development, Year 4, ARGO

Game Design Document

By

Quintin Furlong, Greg Cahill, Keith Wilson, Dylan Curran & Martin Farrell



Influences

- <Baba Is You>

- Video game, puzzle.
- The mechanic of changing properties of on-screen objects using words is a fantastic mechanic to explore, especially combining it with platforming elements.

- <Generic 2D Platformers>

- Games, Platformers (mario, sonic etc)
-The genre as a whole has left predetermined notions in a players mind such as “This is the player” and “This is the goal”. This game completely messes with these and leads to players throwing everything the thought they knew out the window.

- <Pac man>

- One of the oldest and iconic maze games.
- Consistent interesting choices to be made.

The elevator Pitch

A 2D side-view game where the words on screen overwrite logic of the physics as we know it.

Core Gameplay Mechanics (Detailed)

- **<Core Gameplay Mechanic #1>**
  - <Logic Controlling words>

    Words in blocks on screen can alter the logic of the universe. Making other objects behave in ways they normally wouldn't depending on the sentence structure.
  - <How it works>

    The cat can push the words on the screen and move them to another sentence. Altering the logic of the level. If "rock is push" is default, the rock can be pushed around by the cat but if the word "push" was exchanged with the word "you" then the rock has become the player controlled object and responds to player movement controls.

- **<Core Gameplay Mechanic #2>**
  - <Player/Object movement>

    What object the player controls will change during the playthrough of the game. The player's movement ability will be universal across these objects allowing the player to move the object vertically and horizontal
  - <How it works>

    Depending on which object is the current player controlled object the movement script will be called on it and that particular object will have its position vector and sprite's position updated if the player moves.

- **<Core Gameplay Mechanic #3>**
  - <Achievement system>

The game will concurrently keep track of the player's achievements and the player will be able to view these on the achievements menu and/or on the pause menu.

- <How it works>

  When the player achieves something the badge/text or whatever represents it will have some visual indication to indicate to the player that they have achieved that specific achievement.

Story and Gameplay

Story (Brief)

What *really* occurs when you leave your pet at home. Well, we believe that cats and their ability to secretly overwrite the universe to whatever they desire occurs.
Our story follows a young kitten beginning to explore the use of this power in a simple living room...

Story (Detailed)

This kitten named socks begins training its universe altering mind prowess. Of course, these powers are unable to be used when under human supervision so the training can only occur when the human owners have left. The kitten can change what is what, and how it is, but due to it being inexperienced there may be some trial and error involved…

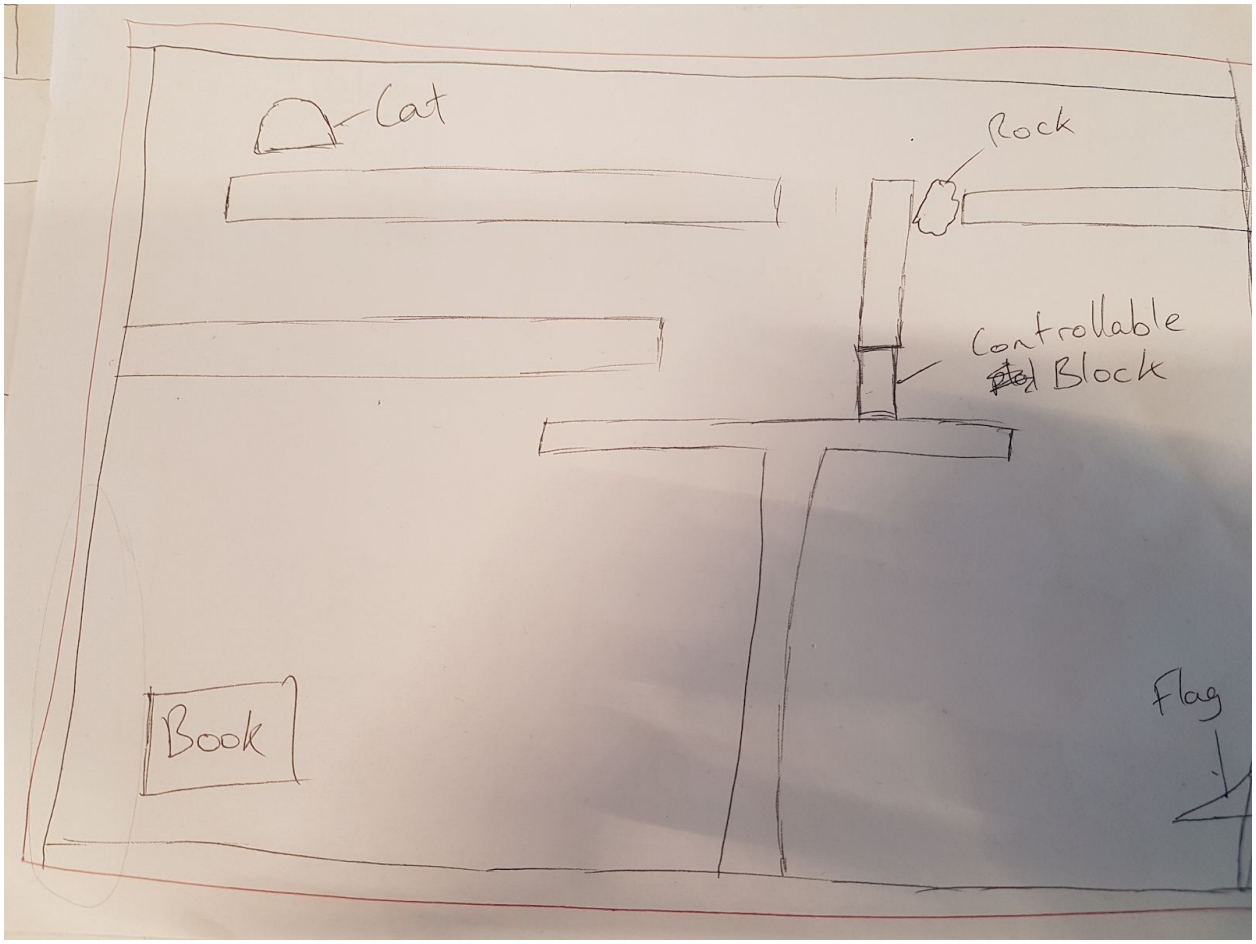
We follow the kitten it controls this small scale environment to explore, slowly expanding its mind to its limitless potential. We wish to reveal the truth of what happens when you leave your pet home alone.

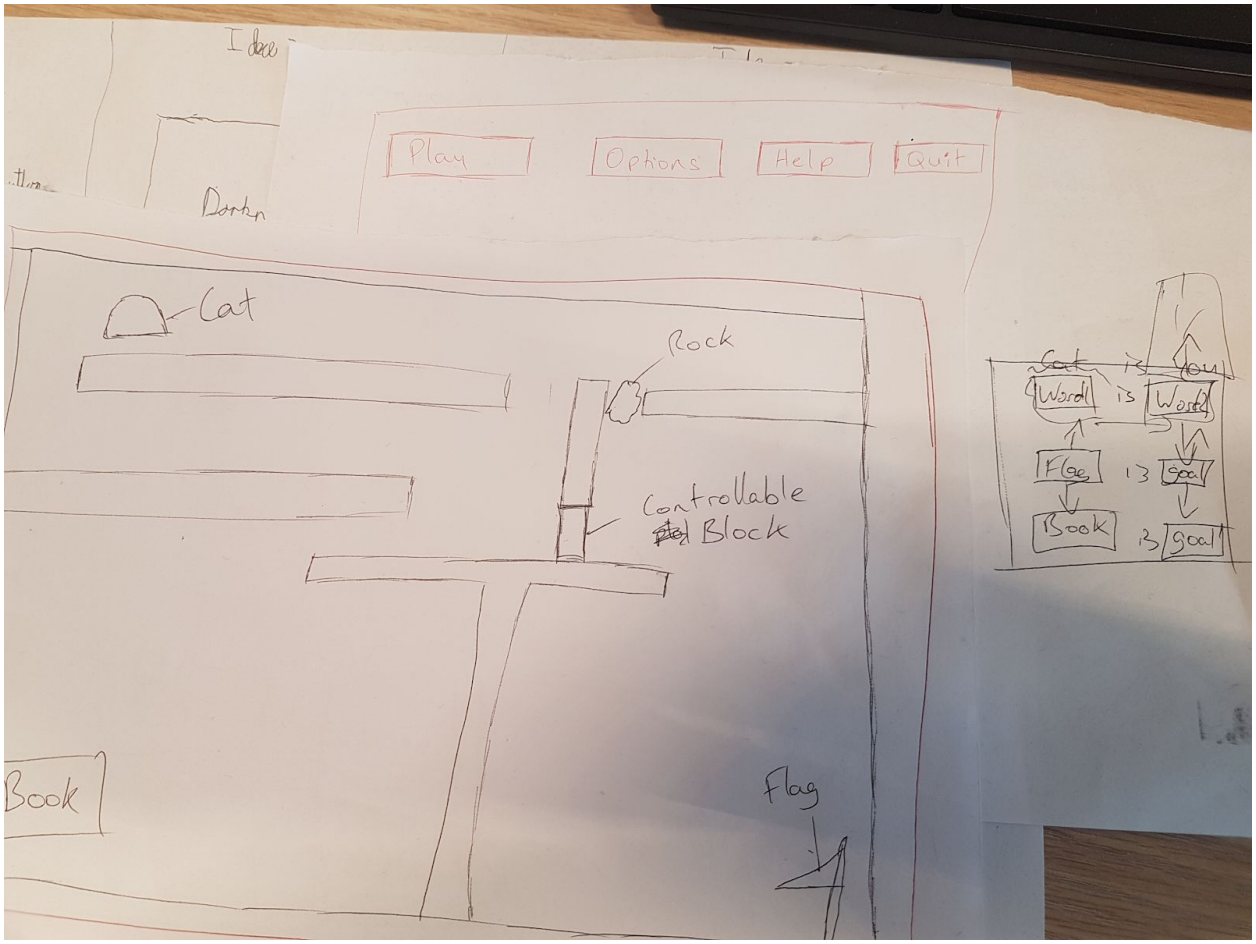
Gameplay

The player can see on screen an average looking level but with the words gravity=up, goal = death, kitten = goal , wall = player. Without the words on screen, the player would probably assume they would control the cat, but when they go to move they move a block (as wall = player) in the wall, then fall up (as gravity = up). Moving around what is what by pushing words, the ‘player’ must navigate to a ‘goal’ whatever either of those end up being.

The player can move the words by using moving in a menu. up and down to cycle through the object list, left and right to change the selected objects properties. The player can rewind one step backwards all the way back to the beginning of the level. Everything else gameplay wise comes from the unique interactions between what currently is and what is not.

Paper Prototypes


Cat
Rock
Controllable
Block
Book
Flag



- 2D

- Textures
  - Maze Wall Textures
  - Goal texture
  - Kitten texture
  - Wall Texture
  - Rock Texture
  - Ball Texture
  - Toilet roll Texture
  - books Texture

- Menu textures: Splash screen, Licence screen, Credits screen and Main Menu background. Menu buttons.

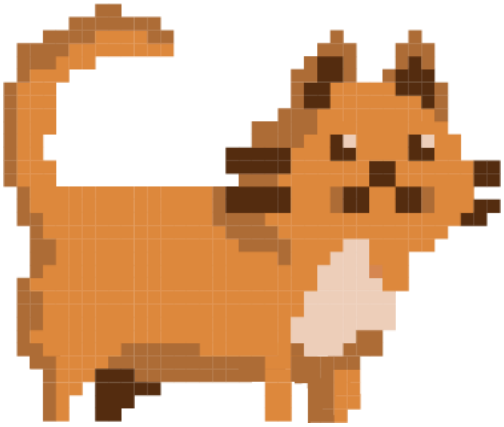
- Heightmap data (If applicable)
    - List required data required - Example: DEM data of the entire UK.
- Etc.

- Sound

- Sound List
    - Background music for game - Background music
    - Cat meowing - Cat sound
    - Cactus hit noise - cactus noise
    - Cat food (passing a level) - Level noise

Features

Title: Cat[GE]



Points : 10

Introduction: The Cat component is what the player originally appears to be and will be in the future, this feature is designed to throw off the player at first and later on the player will become the cat and play as the cat. The idea of the cat feature is that it throws off

the player at first to ensure the player questions the logic of the game, after time the cat will become the player object to be used.

Conditions of Satisfaction for the cat:

- The Cat must be able to move according to what words are applied to the cat.
- The cat should appear at the start of every level.
- The cat should be animated based on what state the cat is in.

Title: Books[GE]

Points: 10

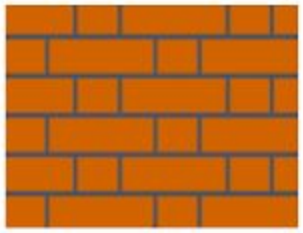
Description:

These will be objects in the game that are not affected by gravity in the game world. The other object can only collide with the top of the platforms.So it allows the player to get to a higher position or to avoid dangers by jumping on top and doesn't prevent the moving object from going up through the platform.

Conditions of Satisfaction:

- Other objects should be able to pass through all but the top side.
- Other objects should be able to walk on it.
- Doesn't get moved by objects in the game.

Title: Walls[RWM]



Points: 10

Description:

These will be objects in the game that are not affected by gravity in the game world. The other object normally can't go through them but can also allow the player to get to a higher position or to avoid dangers.

Conditions of Satisfaction:

- Other objects shouldn't be able to pass through.
- Other objects should be able to walk on it.
- Doesn't get moved by objects in the game.

Title: Floor/Ceiling[RWM]

Points: 10

Description:

The floor will be at the bottom of the visible game world while the ceiling will be at the top. These objects will prevent any objects from leaving the perceivable world either from the top or bottom.

Conditions of Satisfaction:

- No objects can pass through the floor or ceiling and escape the screen.

- The floor will always be at the bottom of the screen and the ceiling will always be at the top of the screen.

Title: Toilet Roll[GE]

Points : 10

Description: The toilet roll component is what can be pushed around initially by the player.

Conditions of Satisfaction for the rock/ball/yarn:

- Toilet roll must be able to be pushed.
- Toilet roll should appear at the start of every level it is placed in.
- The toilet roll should not move through other objects or walls.

Title: Player Controller [GE]

Points: 10

Description:

This feature allows the user to move the object that is currently associated with the user in the game world. As the aim of the game is to get to the “goal”, the associated object completes each level by making contact with the object marked as “goal”.

Conditions of Satisfaction:

- The user input will move the associated object with no viewable lag.
- The feature needs to be usable with every object in the game that can be associated with the user.

- The feature needs to stop working with objects that stop being associated with the user.
- Can make contact with “goal”.

Title: Goal[GE]

Points: 10

Description:

This feature needs to be connected to an object in the world to make completion possible. To complete a level the player has to get the object associated with them to the object associated with “goal”.

Conditions of Satisfaction:

- Should have a possible association with at least 1 object on every level.
- Should be able to make contact with every object that can be associated with the player.
- Should be able to disassociate objects from “goal”.

Title: Credits[RWM]

Points: 10

Description:

This is a screen that appears after completing the final level. It is the word “CREDITS” at the top of the screen on a background. Words start appearing from the bottom of the screen. The words are the name of every one who worked on the game along with the job title they had while working on it. The screen closes once the last name goes off the top of the screen.

Conditions of Satisfaction:

- Opens after the last level is completed with a smooth, non-jarring transition.
- All names should have the same screen time for equality.
- Starts and ends with no names on the screen.

Title: Words [GE]

Points: 20

Feature Description:

‘Words’ are a collection of letters that represent something in the current gameplay section. As an example, Having wall = the property player in the menu where you can change ‘what is what’ means that the actual human player will be controlling a segment of the wall, as opposed to what you believe would be the actual player, and the

objective would be to navigate the level using these ‘loose logic’ logic to reach whatever ‘goal’ may be.

Certain things may be locked as hard set rules in a level to let the player think a small bit more about a solution (example, in a given level it could be unchangeable that kitten is player, but that does not mean that something else can *also* be player)

The conditions of satisfaction:

1. An object being associated with a word will do what is expected (such as something having the player property will move when the human players ‘moves’)
2. the objects interact with each other in expected ways (an object with the player property movinging into an object with the goal property completes the level)

Title: Collision Component[GE]

Feature points: 10

Feature description:

The collision component ensures that if two entities collide together the collision component will handle the event and carry out the appropriate collision actions as a result.

The conditions of satisfaction:

- The collision component should be able to be assigned to any entity within the project.
- This component should handle an event when two entities collide and carry out the correct collisions based on that.

Feature title: Controller Class [GE]

Feature points:10

Feature description:

The joystick class handles the initialization , and closing of a controller once it is plugged into the PC itself. When something needs controller input, a controller variable called Joystick* m_stick will be used (for example, in the game class)

This will be passed to the other states ‘handle events’ to handle button presses/stick movement. The logic of what happens when a press occurs will be kept separate from the joystick class to follow the command pattern

The conditions of satisfaction:

- From any header, #include “Joystick.h”
- create a Joystick* m_stick and then call m_stick.init()
- Check for a button press (on an xbox controller, the following will be the A button)
- if (SDL_JoystickGetButton(stick.getStick(), 0) != 0)
- {
- //Do stuff
- }
- if it enters the function, the controller has been set up and ready to use.
- when closing the game, call m_stick.close();

Feature title: Ability to switch between objects [GE]

Feature points:10

Feature description:

This is the ability for the player to, at run time, set what is a player, what is pushable, what is a goal etc. The Player will be presented with a menu with the default of cat = player, flag = goal, cactus = spiky and so on, the player will be able to select the cat section containing player, and switch the cat to goal, and then go to the flags section and set it to player. Moving will now move the flag, as it is the player and moving into the cat will complete the level as its associated with goal.

The conditions of satisfaction:

- In the gameplay section, Use the dpad to select which word you would like to change.
- press the button that is ‘select’, a drop down menu over the highlighted word will appear with all the possible choices.
- Moving the dpad down/up will switch between the options, and pressing the selector button will confirm the changes.
- Depending on what you the player changed, test out whether it is acting as expected.

Feature title: Movement System [GE]

Feature points:10

Feature description:

The movement system takes a position component of an Entity , and a string (four options, they are “up”, “down” , “left”, “right”)

Passing in up, for example will take that position component, set its last position to its current position then update its position after applying a movement to it.

The conditions of satisfaction:

- In the gameplay section, move the controller stick right
- the object associated with player will move right a set amount
- pressing up will make it move up, and so on
- Pressing the reset button will slowly move back through all the positions back to their original

Feature title: Reverse State Functionality [GE]

Feature points:10

Feature description: This is the ability that all entities, once they move, they will store their last position onto a stack. When the reset button is pressed once, all entities will go back one movement, but if the button is held down all entities will move back to their original state, Irregardless of the amount of moves each entity made, if any at all.

The conditions of satisfaction:

- In the gameplay section, begin moving the player any amount of directions you wish
- With the player entity, move the ‘movable’ entity around as well
- Press the reset button once, the player and moved object will go back to their last position
- hold the button down, The objects will begin moving back to their original positions.

Feature title: Component Class[GE]

Feature points: 10

Feature description: The component class will be a virtual class with functions that can be used and overwritten by each component, like a blueprint.

The conditions of satisfaction:

- When the class is completed and it is possible for another component to inherit from it.

Feature title: System Class [GE]

Feature points: 10
Feature description: The system class will be a virtual class with functions that can be used and overwritten by each component, like a blueprint.
The conditions of satisfaction:

- When the class is completed and it is possible for another component to inherit from it.

Feature title: Splash[RWM]
Feature points: 10
Feature description: The splash screen will be the screen visible to the player between the license screen and the main menu and will simply serve as a transitional image/animation.
The conditions of satisfaction:

- Gamestate can transition from license to splash.
- Gamestate can transition from splash to main menu.
- Sprite/animation draws.
- Splash gamestate is active for 3 seconds.

Feature title: Entity Class [GE]
Feature points: 10
Feature description: The entity class will be a virtual class with functions that can be used and overwritten by each component, like a blueprint.
The conditions of satisfaction:

- When the class is completed and it is possible for another component to inherit from it.

Title: Sprite class[RWM]
Points: 10
Description:

This is used to display the game objects in the game world. It needs to be given the image it should display and other components to set position and size. It can also be used to draw animation for entities.

Conditions of Satisfaction:

- Should display image given.
- Should be positioned and sized correctly based on given data.
- Should be able to use a spritesheet to create an animation.

Title: Words UI[RWM]

Points: 15

Description:

This allows the user to both alter the associations of game objects to attributes like “player”,”goal” and to clearly see all the active associations currently in effect.

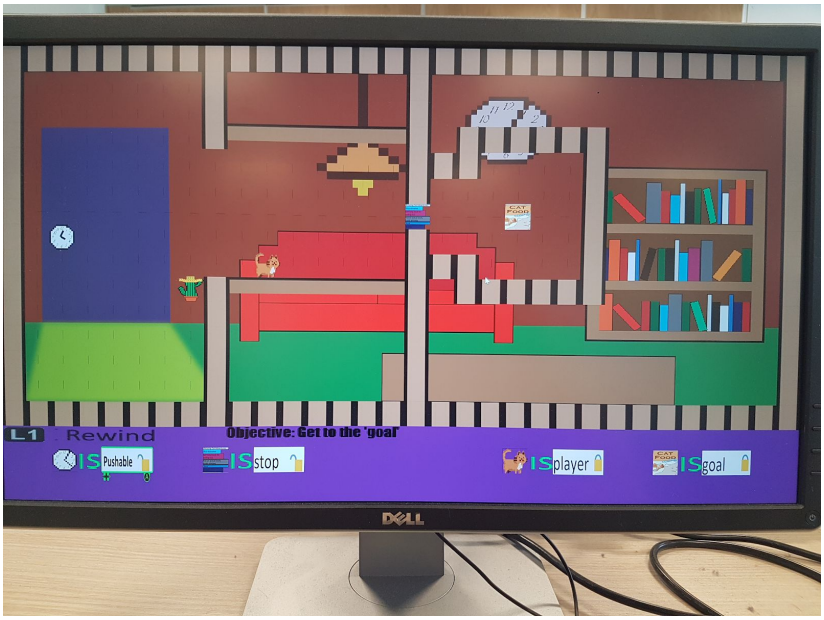
Conditions of Satisfaction:

- The user should be able to make any association that the level allows.
- The changes on the UI should be reflected in the actual game.
- All pre-set rules on the UI need to be in effect at the start of each level.

Title: Level One[RWM]

Points: 10

Description: Level one is the introductory level and should be kept as simple as possible. Limiting the elements to a cactus, a clock, books, a cat and a goal.



Conditions of Satisfaction:

- Should be displayed after pressing play on main menu
- Should be able to get to level 2 using clear logic.

Title: Finite State Machine [GE]

Points: 20

Description:

This is a system that is used to control the transitions between different animations. It will only change to the requested animation, if the previous animation would allow it to happen.

Conditions of Satisfaction:

- Should allow transitions from one animation to another.
- Should prevent transitions that are not set.

Title: Client/Server [OG]

Points: 10

Description:

This is the method that allows players to send and receive data to other players. Each player sends their data to the server and gets others players data from the server.

Conditions of Satisfaction:

- Should be able to connect the clients to the server.
- Should be able to send data to the server.
- Should be able to receive other player data from the server.
- Shouldn't get its own data back.

Title: Level Setup from Files [RWM]

Points: 10

Description:

Firstly the level number is used to determine the correct file for the level. All the data from the file is read into the program. The data is then used to tell what type of object is in each square of the grid.

Conditions of Satisfaction:

- Should accurately display the level that was designed.
- Should be able to display a new level when an old one has been completed.
- The world should also work like it looks, with a wall actually stopping the player for example.

Title: Smooth Movement based off Animation [RWM]

Points: 10

Description:

This has the movement from one square on the grid to another taking place over the course of its animation. An object will start moving at the beginning of an animation and will only get to its destination at the end of the animation.

Conditions of Satisfaction:

- The start and end of the movement needs to match up with the start and end of the animation.

Feature title: Position Component[GE]

Feature points: 10

Feature description: The position component will be a component of the ECS system that stores the position of any entity in a vector and has set and get methods.

The conditions of satisfaction:

- Component has a vector that can be modified.
- Vectors can be obtained via a get method.

Feature title: Velocity Component[GE]

Feature points: 10

Feature description: The velocity component will be a component of the ECS system. It will contain a vector which will be used to represent the speed and direction of an entity. It will also have set and get methods.

The conditions of satisfaction:

- Component has a vector that can be modified.
- Vector can be obtained via a get method

Feature title: Entity Manager[GE]

Feature points: 20

Feature description: The entity manager will house all the references to each entity, keep track of what components each entity has and handle adding and deleting components of the entities.

The conditions of satisfaction:

- It is possible to instantiate and entity.
- Entities can be referenced.
- Components can be added and deleted.

Feature title: Level loader/tile class[RWM]

Feature points: 20

Feature description: Class will be able to take an array of numbers from a JSON file and create and place entities in the level based off of the numbers.

The conditions of satisfaction:

- Class can successfully load JSON file.
- Class can use numbers to load entities.
- Entities are all in their correct places after loading.

- Class only loads certain entities on their respective levels.

Feature title: Main menu[RWM]

Feature points: 10

Feature description: The main menu will be the first intractable menu the player encounters. The player will be able to navigate the buttons that represent play, achievements, help and quit. Play will load the player into level 1 and they can begin to play the game. The achievements menu will display all of the achievements the player has obtained throughout the game. The help screen will display helpful information to help the player learn how to play the game. The quit button will exit the game.

The conditions of satisfaction:

- The main menu is drawn.
- The player can interact with the menu.
- The player has a visual cue highlighting where the cursor currently is.
- The player can press a and execute the buttons on the menu.

Feature title: Boundary System[RWM]

Feature points: 10

Feature description: the boundary system will prevent the player from going outside the bounds of the screen. If the player walks into the edges they will not be able to move any further.

The conditions of satisfaction:

- The player cannot move beyond the bounds of the game.

Feature title: Audio Component / SDL_Mixer[RWM]

Feature points:10

Feature description: We'll be using the SDL_Mixer library to handle all of our sound effects. The audio class will load the audio clips, play them and close them.

The conditions of satisfaction:

- Audio clips are loaded successfully.
- Audio clips can be played.
- Audio clips are managed so they are closed when finished.

Feature title: Scene Manager[RWM]

Feature points: 10

Feature description: The scene manager will handle what scene/level the screen should currently display. This will be an amalgamation of an enum class (representing splash, game, title etc) and the map class which will represent each individual level.

The conditions of satisfaction:

- A “scene” exists for all relevant gamestates/levels.
- Each scene progresses to the next in the correct order.
- There is functionality to go back to previous scenes such as going back to main menu from the gameplay scene.

Feature title: Update on specific gameticks[GE]

Feature points:20

Feature description: Updating physics every 10 frames instead of every single frame will reduce the amount of processing required to play the game.

The conditions of satisfaction:

- Update is only called on entities every 10 frames.

Feature title: Factory Pattern[GE]

Feature points:20

Feature description: the factory pattern uses factory methods to deal with the problem of creating objects without having to specify the exact class of the object that will be created.

The conditions of satisfaction:

- Pure virtual factory class exists.
- Virtual character class exists.
- Subclasses of characters exist.
- Entities are created using subclasses.

Feature title: Command Pattern[GE]

Feature points:20

Feature description: The command pattern encapsulates a request, like an input, as an object. This lets us parameterize other objects with different requests and also greatly reduces the number of computations required. We will use it to get input “requests” which will be strings instead of large switch statements or if statements. The requests will simply be “up”, “down”, “left” and “right”.

The conditions of satisfaction:

- Command pure virtual class exists.
- Request subclasses exist and inherit the “execute” function.
- The execute function will accept a string “request” and the function will then go and perform the code that input calls for(if it’s “up” the character will move up).

Feature title: Scene partitioning[GE]

Feature points:20

Feature description: Scene partitioning reduces the amount of our scene that physics will be performed on. Our game is tile based so the octree and quadtree solutions aren’t really applicable. We decided to only compute our logic within a certain distance of the entity with a “player” component.

The conditions of satisfaction:

- Physics is only performed within a certain radius of an entity with a player component.

- Physics is not performed anywhere outside of this radius.

Feature title: AI[AI]

Feature points:50+

Feature description: The AI will have various components and functionality to it. For all intents and purposes it is a bot controlled player that will move through the level using field flow pathfinding and make various decisions using a behaviour tree.

The conditions of satisfaction:

- AI exists.
- AI can move.
- AI can pathfind.
- AI can make a decision from the behaviour tree.
- Decision is selected at random but does not perform the same behaviour more than once.

Feature title: Achievements[GE]

Feature points: 20

Feature description: Players will get achievements for doing various things in the game such as completing levels or dying.

The conditions of satisfaction:

- The achievements are achieved when specific conditions are met such as dying to a cactus.
- The achievements menu is updated to display the achievements you have obtained.
- There is a small visual indicator to give the player some feedback about the achievement in the game.

Feature title: Help[RWM]

Feature points:10
Feature description: This screen will simply display instructions on the screen for playing the game and may also have a back button to return to the main menu.

The conditions of satisfaction:

- The instructions are displayed on screen.
- The back button is displayed on screen.
- Clicking on the back button returns the player to the main menu.

Feature title: Level Two[RWM]
Feature points:10
Feature description: Level two will still be quite simple but will introduce an additional mechanic to the puzzle. Level one it is possible for the player to learn that cacti are bad and that you can die to them. On this level we will reinforce this concept to that it is pretty much guaranteed the player will know for future levels. In order to achieve this the puzzle will be designed in such a way that the player is forced to die to the cactus in order to complete the level.

The conditions of satisfaction:

- The level is loaded from a JSON file.
- The objects are updated correctly at run time.
- The puzzle can only be solved by dying to the cactus.

Feature title: Level Three[RWM]
Feature points:10
Feature description: Level 3 is a continuation of the new learning from level 2 where the player must take control of another object in the game which can die to the cactus to clear the way for the cat to get to the goal.

The conditions of satisfaction:

- The level is loaded from a JSON file.
- The objects are updated correctly at run time.
- The puzzle can only be solved by dying to the cactus.

Feature title: Level Four[RWM]

Feature points:10

Feature description: the player must navigate a small maze and destroy a cactus with a book or push toilet roll into the cactus to pass the level. The cat can then approach the goal.

The conditions of satisfaction:

- Level 4 is loaded from a JSON file.
- The objects are updated correctly during run time.
- The puzzle is solved if the cat is the object to collect the cat food.

Feature title: Level Five[RWM]

Feature points:10

Feature description: Level three will really up the difficulty level and make use of 20+ pushable objects to create a puzzle that should take quite some time to solve and force the player to use the rewind function once some element of the puzzle becomes “unsolvable”.

The conditions of satisfaction:

- Level three is loaded from a JSON file.
- The objects are updated correctly at run time.
- The puzzles incorporate 20+ pushable objects that will most probably cause a player to use the rewind function.