APOCATLYPSE MEOW Design Document

GAME OVERVIEW

Game logline

ApoCATlypse Meow is a 2D puzzle-platform game where cats are the only ones that can save mankind from an army of robots that want to bring order and cleanliness to the entire world.

Gameplay Synopsis

ApoCATlypse Miao is a platform game that includes puzzle element. If the player jumps from an high place he/she can flip the gravity, this way player's character experiences a flip in the gravity direction that results in the ceiling becoming the new ground. Only the player's character is affected by this Gravity Flip.
Player can chose among different characters, all cats, each one with its own unique features. All of them can jump, climb, crouch, dash, and flip gravity.
Each level contains some collectibles, represented by hamburgers, however the player does not have to collect them all to finish a level.
The setting is a super ordered post apocalyptic world, where humans are absent from all the environments, leaving place for robots and cats.
Look and feel of the game combines the desolation of a post apocalyptic world with the maniacal order and cleanliness enforced by robots.

Target Audience

At the moment we are not planning a public launch, however, we adopts an explicit approach, rather than a subtle one, when dealing with indications and tips to the player. This is done since the game must be playable by any kind of user.

Platform

The game will be implemented considering PC platform as target; due to its 2D graphics we expect low minimum requirements.

DETAILED DESIGN

Character Design

All characters that have to be animated must not have arms or legs. Their feet, hands and head must be separated from the rest of the body. Further details in Pipeline Overview.

(Only the first two kittens out of four will be developed for the Course Demo)

Kitten 1: Great
Description: young Caracal confident in his heroic allure.
Signature power: he is the only cat who can jump twice.

Kitten 2: Giza
Description: cute female Sphynx with big intelligent eyes.
Signature power: she can pass through wet floors, showers, filled sinks etc... and also on fragile surfaces.

Kitten 3: Ton Guy
Description: retard Scottish fold, with the tongue always out of the mouth and big, gentle eyes. Pretty round and big. Purple ribbon tied on his tail. He is a fail-jump cat.
Signature power: he start swirling faster and faster, creating a Kitten Tornado capable of destroying some heavy obstacles and robots.

Kitten 4: Johnson
Description: screaming cat (Oh Long Johnson Cat); Long hair, black back, white belly, white feet. His facial expression is always grumpy.
Signature power: tape a pink jam toast on its back, jump, start swirling on itself, inhale deeply and then transform to become the Nyancat, https://www.youtube.com/watch?v=QH2-TGUlwu4 (super-speedily flying toward, and temporarily invincible). With this, he can cross chasm, water pools, lava and other dangerous terrain too big to be jumped.

Mice Emperor
Description: ancient Roman-clothed white mouse (the typical lab guinea pig) with pink eyes and thick geek-like glasses, sitting on a throne made of a pro PC-gaming mouse.

Basic Robot
Description: one head, two hands with few fingers, two legs and one unique central block as body, the design should be as generic as possible. We want the player to focus on the kittens, not on robots.
Behavior: it runs towards the player and tries to capture him/her.
This kind of robot has two variants:

- with cleaning apron
- with chef hat and knives (kitchen robots)
- with oriental traditional clothes and knives (kitchen robots in Zen Restaurant)

Gravity Robot (also called Flip Robots)
Description: similar to Basic Robot but with a scraper (“benna” in Italian) instead of its hands.
Behavior: it run towards the player and tries to throw him/her upside down, then a Gravity Flip is automatically triggered.

Laser Robot
Description: it has no leg, since it is stuck in the ground or on the ceiling and cannot move.
Behavior: it shoots lasers from its position to a certain direction. It may be sleepin and then wake up after some action performed by the player.

Drone Robot
Description: it has no leg, it uses engines on its hands and/or heads to move the fans that keep it mid air.
Behavior: it moves back and forth patrolling a certain area, or it may stay still.

Types

Playable Characters:

- Great
- Giza
- Ton Guy (not developed for the Course Demo)
- Johnson (not developed for the Course Demo)

Non Playable Characters:

- Mice Emperor
- Robots:
    - Basic robots
    - Gravity robots
    - Laser robots
    - Drone robots

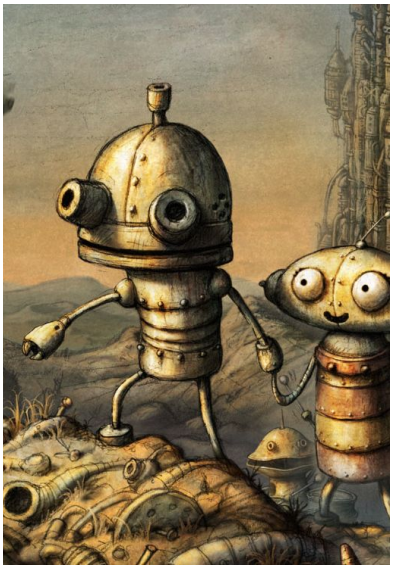
ArtBible

All characters that will be animated:

Great: based on the Caracal species https://en.wikipedia.org/wiki/Caracal

Giza: based on the Sphynx species https://it.wikipedia.org/wiki/Sphynx

Inspirations for **robots**:



Game Mechanics

- **Walk:** player moves at normal speed.
- **Run:** player moves faster.
- **Jump**
    - **from ground:** player moves vertically far from ground (or ceiling if gravity flipped)
    - **from wall:** player moves far from a wall, or far and then back to the same wall. In both cases there is also vertical movement far from ground.
- **Dash:** player moves way faster in straight direction for a less than a second.
- **Crouch:** player lowers to fit into small spaces.
- **Gravity Flip:** player is affected by gravity opposite in sign with respect to all other objects. In order to activate this mechanic the following conditions need to be meet: the player is mid air and the height from the closest ground under him/her is above a certain threshold, which is called *Minimum Flip Height* in Scene Parameter section of this document. After being activated, it lasts for a limited amount of time defined by *Seconds upside down* parameter in Scene Parameter section of this document.

Interfaces

When inside a level, the player sees an icon and a number which represent the amount of remaining lives. The initial amount is 9, since cats are said to have nine lives. (Source: “According to a myth in many cultures, cats have multiple lives. In many countries, they are believed to have nine lives, but in Italy, Germany, Greece, Brazil and some Spanish-speaking regions, they are said to have seven lives, while in Turkish and Arabic traditions, the number of lives is six.” https://en.wikipedia.org/wiki/Cat#Superstitions_and_cat_burning)

Controls

- **Walk:** press *A/D*, or *left/right-arrow*. By keeping it pressed the player can walk without interruptions.
- **Run:** while walking keep *Z* or *left-shift* pressed to move the character in the same direction of walking but at higher speed.
- **Jump**
    - **from ground:** press *space* or *up-arrow* to jump vertically far from the ground (or the ceiling if a Gravity Flip has been performed)
    - **from wall:** press *space* or *up-arrow* to slide vertically on the wall; press *space* or *up-arrow* and *A/D* or *left/right-arrow* to either jump from the

current wall back to the same wall, but in a higher position, or jump away from the wall, on ground or on another object.

- **Dash:** press *X* to instantly accelerate the movement in the same direction of the walk/run for a limited amount of time.
- **Crouch:** keep *left-ctrl* pressed to lower the character and allow it to move under some objects. If the key is pressed while the character is already moving then in addition to lowering the character a dash in performed too.
- **Gravity Flip:** press *F* to reverse gravity only for player character if activation conditions defined in Game Mechanics section of this document are met.
- **Pause:** press Esc

Levels

(Only the first two and the last environments will be developed for the Course Demo)

- **The Rich Aunt House:** an old-fashioned, baroque western house interior, with lots of ornaments and nice porcelains.
- **The Zen Restaurant:** a sophisticated eastern restaurant with jade trinkets, bamboo sculptures, precious Ming jars and fancy waterfalls, pools and water games, terrifying most kittens.
- **The Carpet Factory**: lots of colored carpets piled up and hanging from the ceiling, on which the kitten can climb with his fangs, ruining them.
- **The Military Barrack**: robots are alerted. They know kittens are coming, and wait for them wearing combat arms, using flying drones, closing vault doors and firing bullets and lasers.
- **Overworld:** an external hub connecting the previous levels.

TECHNICAL DESIGN

Pipeline Overview

Animations will be implemented using Unity (https://youtu.be/4qE8cuHI93c?t=313), so for each character the different parts of the body needs to be provided separately (i.e. not overlapping one another), in one unique file.

System Limitation

None at the moment. While coding, where possible always remove all *GetComponent*, *FindObject* and *new* variables declarations from *Update* and *FixedUpdate* functions, and put them in *Start/Awake* or in functions that are not called once per frame.
Expected screen output: 16:9. Final resolution will be decided later in the development, while different aspect ratio might be considered in the future.

Scene parameters

Player

- Mass: 1
- Gravity scale: 2
- Minimum Flip Height: 3
- Layer Colliders to be considered: Ground
- Seconds upside down: 5

Camera:

- Projection: Perspective
- Field of View: 60
- X Margin: 0
- Y Margin: 0
- X Smooth: 20
- Y Smooth: 20
- Max X: 1000
- Max Y: 1000
- Min X: -1000
- Min Y: -1000
- Rotation Angle: 5
- Rotation Time Scale: 0.5

Media

Sprites

- Cats:
    - Great gameplay version
    - Great cutscene and dialog version
    - Giza gameplay version
    - Giza cutscene and dialog version

- Robots:
    - Basic robots
    - Basic robots with cleaning apron
    - Basic robots with chef hat and knives
    - Basic robots with oriental traditional clothes and knives
    - Gravity robots, also called flip robots.
    - Laser robots
    - Drone robots

- Objects:
    - Portrait
    - Sofa
    - Books
    - Chandelier
    - Chimney
    - Clock
    - Column
    - Green curtains
    - Heart
    - Lamp
    - Large table
    - Library
    - Library small
    - Armchair
    - Picture1, 2, 3
    - Rocks
    - Shelf
    - Shelve
    - Sofa
    - Stairs
    - Stairs flipped
    - Table
    - Tree
    - Trophy gold
    - Trophy silver
    - Trophy bronze
    - Vase
    - Window
    - Wood edge
    - Cheeseburger

Music

- Main menu music
- General background for house level
- Cat spotted / action music

Sounds

- Click menu button sound (might be a “Meow”)
- Cat related sounds:
    - various “Meow”
    - paw sound for walking, running,...
- Robots related sounds:
    - moving fans
    - laser noise
    - robot mechanical robots
- Breaking objects:
    - Vase

Asset folder structure

TO BE FINALIZED

Only main folders are listed here (ie those folder containing files that are likely to be highly involved during development).

_Animations:

- **Animation_Franco**
- **Animation_Luke**

_Materials:

- LaserMaterial: material used by the laser line renderer component.
- MissileMaterials: materials used by missile particle systems.
- Others

_Prefab:

- **DialogueSystem**: see details in Dialogue System section of this document.
- **Laser**: see details in Laser System section of this document.
- **Meshes**
- **Missile**: see details in Missile System section of this document.
- Others

_Scenes:

- **Edoardo_Scenes**: testing scenes.
- **Final_Scenes**: scenes that will be in the game.
- **Franco_Scenes**: testing scenes.
- **Luke_Scenes**: testing scenes.

_ScriptableObjects:

- **Dialogues**: contains all the sentences, speaker names and reference to speaker images for all the dialogues of the game.

_Scripts:

- **DialogueSystem**: all the scripts powering the Dialogue System.
- **LaserSystem**: all the scripts powering the Laser System.

- **MissileSystem**: all the scripts powering the Missile System.
- **Script_Edoardo**: scripts written or modified by Edoardo.
- **Script_Franco**: scripts written or modified by Francesco.
- **Script_Luke**: scripts written or modified by Lukasz.

_Sprites:

- DialogueSprites: folder containing folders which contain Great's, Giza's,... sprites to be used during dialogues.
- LaserSprites
- MissileSprites
- Others

Dialogue System

All basic objects needed to show a dialogue can be found in _Prefabs/DialogueSystem.
Each scene must contain a *DialogManager* object, a *ScreenCanvas*, and an *EventSystem*. If an *EventSystem* is already in the scene there is no need to add a new one.
NOTE: Without *EventSystem* no input will be received by the UI.

ScreenCanvas contains five canvas: one for the Main Dialogue, one for the Minor Dialogue, one for the Floating Dialogue, one for the Tip and one for darkening the background during main dialogues. The implementation of the Dialogue System relies on *ScreenCanvas* tag and hierarchical structure. This means that changing the name of any of the objects contained within *ScreenCanvas* will break the code (lines ~90-125 of *DialogueManager.cs*).

In the inspector of Dialogue Manager it is possible to set five parameters: the first two are the word(s) that will be shown on the button used to continue during a Main Dialogue. It is highly recommended to use different word(s) when the dialogue ends, so that the player knows he/she will be back in action. Next two parameters provide the alpha values for the dark background of a Main Dialogue: how much it is increased each frame, and which is the alpha threshold that stop alpha increase. Note that the gameplay is immediately stopped as soon as the dialogue is triggered. Same for resuming gameplay: first the alpha is decreased, then the flow of time is restored. Last parameters is the scaling factor used during animations of all dialogue UI elements, both when entering the screen and when leaving it. It is suggested to use a value that can be perfectly summed up from 0.0f to 1.0f and viceversa.

Three kinds of dialogues that can be used on one of the four possible outputs.

MainDialogue, which is a dialogue that stops time, instantaneously hides any other pre existing dialogue UI, darkens the background and can show multiple speakers one after the other. To move from one sentence to the following one, the player has to press a button. What is shown on that button can be set in the Dialogue Manager Inspector.

MinorDialogue, which is made of only one sentence said by one speaker and does not stops time. It disappears after a given amount of time. It can be shown either on the bottom of the screen, including speaker name and picture, or in the floating cartoon above character's head. In order to activate the latter, the *Is A Floating Dialogue* parameter must be activated in the dialogue inspector.

Tip, which is used to inform the player about what he is supposed to do, or to focus his/her attention to something. It does not stop time and by default is persistent until a Main Dialogue is displayed, or the player enters a trigger that deactivates the Tip, or *Hide Tip After N Seconds* is enabled. It is possible to show only a Tip, or to show a Main Dialogue/Minor Dialogue that includes a Tip.
In any case if a new Dialogue that has the same output of a current existing one is played, then the old one is overwritten.
To show a dialogue, put a *DialogueTrigger* in the scene; to hide a Tip, or a Minor Dialogue, or a Floating, or any combinations of the previous ones, use *DialogueKiller* and specify the *dialoguesToBeKilled* in the Inspector, or use *DialogueTrigger* and provide a Main Dialogue object. If a *DialogueKiller* is used, then it is possible to specify whether the UI should animate or whether it should be hidden immediately without performing any animation. If a *DialogueTrigger* is used to show a Minor Dialogue, then any pre existing Tip will be left untouched.
Triggers that require a dialogue objects as input accept both Main Dialogue, Minor Dialogue and Tip Dialogue objects as valid input. All these three types of Dialogue can be created via Unity menu *Assets*/*Create*/*Dialogue System*.

Laser System

The LaserSystem prefab can be found in the Laser folder inside the Prefab folder. To use that system, drag and drop it inside another object. Note that there are a list of limitations that must be respected, otherwise the system will not work or may not work properly. The list can be found at the end of this section.
In addition to the prefab, there is one last object: the *ObjectInSightLaserTrigger* script that can be found inside the *Script*/*LaserSystem* folder. This script can be put on a object and used to check a certain direction for certain colliders.

ObjectInSightLaserTrigger checks if any object belonging to one of the layers in *Layer To Be Considered* is hit by the raycast starting from *ObjectInSightLaserTrigger*

position and moving toward *Sight Direction*. If this is the case, a laser from the *Starting Position* of the provided *Laser System* is shoot in direction *Laser Direction*, form a certain amount of time. It is also possible to define if the pre-laser cone animation should be performed.

When positioning the system, only the positions of *LaserSystem*, *LaserBeam* and *LaserEmitterStart* will be take into account. The other objects are moved at runtime based on the positions of the previously listed ones. It might be better, just for consistency, to keep all position to zero and move only *LaserEmitterStart* to the desired position. You should position the system when the parent object is scaled 1, and not -1, in order to ensure appropriate behavior.

In order to work correctly, *LaserBeam* and its son objects belong to *Ignore Raycast* layer; *LaserBeam* script in *LaserBeam* must have all layer except *Ignore Raycast*, as activated in *Layers Colliders To Be Considered*. Here it is also possible to provide a *Line Width* for the laser beam.
Laser Collider Script has only one boolean parameter: if true then the laser beam is deactivated as soon as it touches the player colliders. In the script code there is a *TODO* that can be substituted by the code that need to be executed when the cat is hit by the laser.
The *Edge Collider 2D* must have “Is Trigger” property active.

Inside *LaserSystem* object *Laser System* script can be used to set the parameters for the pre-laser cone animation. The *Starting Angle* defines how far from the final laser direction the initial position of the cone edge must be. The *Degrees Per Update* defines how many degree the edge should be moved during each frame. To modify the size of the edge simply update the scale values of *LaserConeEdgeOne* and *LaserConeEdgeTwo*. It is strongly recommended to keep these values equal between the two objects.

To test the laser the *LaserTestingScript* can be add to the parent object. *Laser Direction* and the amount of Laser *Time* before being automatically deactivated can be set in this script parameters, in addition to a boolean value that states whether the pre-laser cone animation should be performed. The *Activation Trigger* can be activated at runtime to activate the laser.
In the **final game**, a script in the parent object will perform the same operation of the script: retrieve instance of the *LaserSystem* and call *ShootLaser* method. If you want a laser that never ends you can provide *Mathf.Infinity* as *laserTime*.
LaserSystem retrieval can be hardcoded in the *Start()* relying on *tranform.Find()*, or it is possible to define a *public LaserSystem* variable so that the laser system(s) can be provided via drag&drop in the Inspector. In the first case if the same parent has

multiple *LaserSystem*s then you should rename each one and then use the different names to retrieve the different objects.

List of limitations:

- The laser beam can be shot only if there is something to hit. If the raycast hits nothing, then no laser will be shown.
- *LaserSystem*, and an object which uses the *ObjectInSightLaserTrigger* script, should have only one object as parent, not a parent with another parent and so on. This is due to the fact that only the scale property of the parent object is checked; this was done in oder to make sure that the code moving the enemy will change the scale of that particular parent object. It is possible to remove this limitation by adding a parameter in inspector where the object, whose scale will be changed, is put. But I am afraid that if then the enemy script moves another objects, it could be very difficult to find where the problem is.
- Position *LaserEmitterStart* only when the scale.x of the parent object is 1, or positive, to ensure appropriate behavior at runtime.
- If a Main Dialogue starts, the pre-laser cone animation does not stop. The code can be updated to get the correct result, if needed ask Edoardo to modify the Update function of Laser System. Same for slow-mo during Gravity Flip. The pre-laser cone animation does not slow down.
- The laser beam moves as the parent object moves, but DO NOT ROTATE THE PARENT object. This would cause the edge collider of the laser rotate, while the laser beam stays still since the laser direction is fixed. The result would be a collider no longer aligned to its visual counterpart. As long as the parent object moves only horizontally and vertically without rotating it is not a problem.

Missile System

In *Prefab/Missile* there are three prefabs: *Missile*, *MissileLauncher* and *MissilePoolManager*. Each scene where at least one missile is shot must have one and only one *MissilePoolManager*. On each shooting spot one *MissileLancher* must be put. The *Missile* prefab must not be add to the scene.

MissilePoolManager parameters are: how many missiles are instantiated automatically when the player enters the level, in case of need new *Missiles* are instantiated; and the *Missile Prefab* variable which must contain the *Missile* prefab that can be found in *Prefab/Missile*.

MissileLauncher is where the parameters of all the missiles launched from that spot are configured. Details about each parameter can be found by leaving the mouse on

the variable names in the Unity Inspector. The long list of parameters allows us to: shoot a certain number of missiles directly to the player by providing their number, the amount of seconds between each launch and the amount amount of seconds between each serie of launched; shoot a missile on a fixed direction for a certain amount of time and then start following the player; shoot the missile with a different initial speed than the hunting speed and provide how many seconds are needed to perform the linear transition from *Launching Speed* to *Hunting Speed*; add to player's rigidbody an explosion force whose direction can be the natural one with respect to the point where the missile hit the player, or a fixed one that is provided via Inspector, and in this case it is also possible to say whether, if Gravity Flip is active, the direction should be flipped as well; lastly, we can define which layers of colliders are taken into account for explosion. My (Edoardo's) suggestion is to keep all layers enabled except *Ignore Raycast* and *Missile*. If you want the missiles to explode if they collide with one another then it is sufficient to enable the *Missile* layer.

Note: no missile is launched if *Time Between Series Of Launches* seconds have not passed since the start of the level.

Missile prefab *Missile rotation coefficients* should not be modified unless the missile rotation during flight is not the desidered one. In this case the *Direction Variation Multiplier* parameter should be modified. *Explosion Sound* parameter can be used to change the sound the missile reproduces during the explosion. To change the fire or the explosion please modify the respective particle system. The script expects the *Missile* object to be not active, the *FireParticleSystem* to be active and the *ExplosionParticleSystem* to be not active. If these conditions are not met then the code will not work properly.

Miscellaneous notes for programmers

- "eg" in front of an object in the assets folder means that that object is a dummy created for local testing; it should be removed before shipping the game.
- PlayerPlatform updates in the following commit https://github.com/Lykos94/ApocatlypseMiao/commit/6e540962185b6485d96a2b8f0f604e08f8d1ec10 may prevent that code from working properly if player's friction is not constant.
- For **performance** reasons, whenever possible avoid using GetComponent, FindGameObjectWithTag, other game objects retrieval methods, or creating new variables in either Update or FixedUpdate methods. Perform those operations in Start or in a method that is not called once per frame.
- Robot clothes will be children of the parent object.

STORY

Tagline

A cute-speaking, charismatic kitten rallies a team of turbulent cats to escape the grasp of order-compulsive robots and save humankind from slavery.

Game Intro

At some point of history, humanity lost.
Now an army of robots clean empty houses, restore art pieces no one will ever admire, and cook food no one will ever eat.
No one.
Because right now billions of people are being kept somewhere, suffering. Robots don't care: all they care about is order, safety and rationality. They have no fear of humans: they're far, far smarter than them.
*But now, here stands a hero who doesn't give a s**t about how smart those iron jackasses are.*
Because he's a cat.
*And cats don't give a s**t about anything. Ever.*
Be prepared, robots: apoCATlypse is coming.
MEOW

Short Intro

Humanity lost.

Now, billions of people are kept somewhere, suffering. Robots won the war because they were very much smarter than humans.

*But now it comes a hero. Someone who doesn't give a s**t about how smart those iron jackasses are. Because he's a cat.*

*And cats don't give a s**t about anything. Ever.*

Be prepared, robots: apoCATlypse is coming.
MEOW

Game Setting and Story

This post-apocalyptic world looks like the dream of a person affected by obsessive-compulsive personality disorder: nice vases are on top of shelves, height-aligned shiny bottles and color-ordered ball of thread. Of course, this tidy world can not coexist with a bunch of skittish and lively kitten like you, and this will enrage the featureless robots army which is keeping everything the way it is. Only one distinguishable mark on these iron creeps is the mysterious writing “M. Empire” on the belly; you don’t know what this means, and care even less. You only care about freeing your kitten friends and, eventually, save your favorite race of servants and worshippers, the humans.
After the first team member Giza is rescued, you’ll discover who is actually behind the robots. They’ll end up being their ancient enemies: mice! (M. Empire means “Mice Empire”)
Mice Emperor is an ancient Roman-clothed white mouse (the typical lab guinea pig) with pink eyes and thick geek-like glasses, sitting on a throne made of a pro PC-gaming mouse. Freed himself from human experiments some time ago, he got revenge on his persecutors unlashing IA-controlled robots against them, which captured every man, woman and child in huge subterranean mazes like the ones scientist used to put him. The robots then moved on removing every other “imperfection” of the world.
After meeting him, the Mice Emperor will try to convince you to give up your crusade, being humans cruel and deserving to be put in a cage; but you are the kitten hero, and won’t listen him.
Because he speaks difficult and you lack focus.
Then, you and Giza will cooperate to defeat his first boss robot, a giant iron beast.

Characters and relations

(Only the first two kitten will be developed for the Course Demo)

Kitten 1: Great (main character)
Personality: Always energic, brave and cocky; he loves heroic poses and boastful statements even if his lolcat-language speeches are incoherent because his total lack of focus. He often jumps form one topic to another in a dumb and funny way, and he’s totally unaware of this. He’s confident about his heroic allure, but his speeches usually ends up to be stupid or at least very naive. https://www.youtube.com/watch?v=cNycdfFEgBc
Signature Power explanation: no one can jump twice in the air, everybody knows. But he doesn’t know that, so he can.

Kitten 2: Giza
Personality: Smart, calm and brave, she is the most “human” of the four kittens. She is the one doing plans other cats ends up screwing up, and is the only one always talking correctly. She is pissed of by stereotypes, and gets angry when the main character act like a typical internet lolcat.
Signature power explanation: As a Sphynx, she doesn’t fear water, and can thus pass through wet floors, showers, filled sinks etc… . She’s also very light, and can pass on fragile surfaces.

Great-Giza relation: Great likes her, but is embarrassed by her lack of hair which he find too sexy and provocative, like she is going around naked. She call him idiot for that.

Kitten 3: Ton Guy
Personality: Good hearted, love being pet, altruistic and loving. He hates the purple devil always following him (the ribbon on his tail).
Signature power explanation: trying to catch his own tail, he start swirling faster and faster, creating a Kitten Tornado, actually able to destroy some heavy obstacles and robots. https://www.urbandictionary.com/define.php?term=kitten-tornado

Kitten 4: Johnson
Personality: Cynic and always unimpressed, the only thing he likes in the world is to sing, and he fight to restore Opera Houses – but when actually singing, his voice it’s weird and babbling like in the video, and nobody understand his songs’ lyrics. https://www.youtube.com/watch?v=kkwiQmGWK4c Everytime the main Kitten ask everybody to “give me meow!” he answers: “No, because when I say that word, I kind of lose control.”
Signature power explanation: Tape a pink jam toast on its back, jump, start swirling on itself (like on this commercial: https://www.youtube.com/watch?v=XMW_bA9ilKo), inhale deeply and then transform to become the Nyancat, https://www.youtube.com/watch?v=QH2-TGUlwu4

Script

Legenda

Bold parts: are meant to be captions, indications for the player.
(Italic parts): Those are triggers and things I wrote for the team only.
Normal parts: Line of dialogue of the characters

Lαtβɣ: dialogue id; α is the level number, t is the type of the dialogue (ma for Main, mi for Minor, fl for Floating, ti for Tip), β is the sequential number of the dialogue, ɣ is the speaker name identifier (GR for Great, GZ for Giza, GR-GZ for both Great and Giza and so on).

Level 1 – Rich aunt's house

GREAT: I wuz away 2 weekz. Meow Im hungry!
Find and enter rich Aunt's house

GREAT: Iz dat old hoomanz houz? Lot ov fud evrytime!

L1mi1GR
GREAT: Im bak, old hag! Iz ther a cheezburger 4 me?
Jump on the chair.

L1mi2GR
GREAT: Not here. Maybe on dat table? Double jump tiem!
Double jump on the table.
(When jumping there, he broke a vase)

L1mi3GR
GREAT: Oh I broke dis. Sry.

L1mi4GR
GREAT: Nothin her! Meowbe in da cellar?
Jump on the chandelier and push the right border to make it sway.

(When he reaches right border of the house, a sleeping robot is sitting on the cellar door)

L1mi5GR
GREAT: Nowai! Wuts dis ting sittin on cellar door?
Iz closd! Wer hoomanz can be?
Explore other floors. You can cling on vertical surfaces.

(after he reached intermediate floor clinging)

L1mi6GR
GREAT: Hay, hoomanz! I did dat ting wif mah nails! U mad?

L1mi7GR
GREAT: Come on! I has been away only 3 weekz.

L1mi8GR
GREAT: Hoomanz! Wer r u? Im hungry!

(after breaking several lined objects)
L1mi9GR
GREAT: Mew, iz pritee funny 2 do dis… ehe…

L1mi10GR
(On the top floor there's a cardboard box with a cheeseburger logo)
GREAT: Whoa! Up ther!

(Under the box is an alarm button. When the cat messes with the box, Robot activates in all the house including lasers; music changes)
L1mi11GR
GREAT: Mew! Why metal cans r movin?
Cellar is now accessible. Reach it!

(When taking the first damage, or wherever you find it fitting)
L1m12GR
GREAT: You cat to be kitten me!

Meeting Giza

L1ma1GR-GZ
GIZA: You, there! Can you free me, please?
GREAT: (*heart-shaped eyes*) Whoa! U r gurl kitteh!
Can I spend mah 9 livez wif u?
GIZA: What? But you don't even know me!
GREAT: No need, u r so beaufiful!
GIZA: That's not even remotely an explanation.
And why on earth are you talking like that?
GREAT: Like wut?
GIZA: Like a silly internet lolcat.
GREAT: Doan knoe. Im out ov focus.
GIZA: That has nothing at all to do with… well, nevermind.

This cage is locked, can you set me free?
GREAT: Uh, wait... U r all naked! Robots stole ur hair?
GIZA (*blushing*): I was born like this, you dummy!
I'm a sphynx cat. My name is Giza.
GREAT: My naym iz Great. Im a hero!
GIZA: A hero? Yeah, for sure.
GREAT: Jus wait, Giza. I'll set u free!
GIZA: I'm... not going anywhere.

After releasing her

L1ma2GR-GZ

GREAT: Hay, Giza! I maid it, ur free!
GIZA: I hardly believe it.
GREAT: Teh lil red devil on ur cage vanishd.
GIZA: Little red devil… you mean the cage's electric lock?
You turned it off? How did you do that?
GREAT: Meow, I doan knoe, I doan care. R we off?

TUTORIALS

At the beginning of the first level, the tutorial will show up. It will explain the following mechanic:

- Movement basics
- Jump
- Double Jump
- Dash
- Crouch

After that, at some point of the level the player will be explained the main mechanic: Gravity Flip, as it is necessary to overcome certain points and challenges.

PROJECT SCHEDULE

Main deadlines:
End of November: first level completion.

Weekly deadlines: see *Homework* section in meetings pdf inside *Meetings* folder on Github project repository.