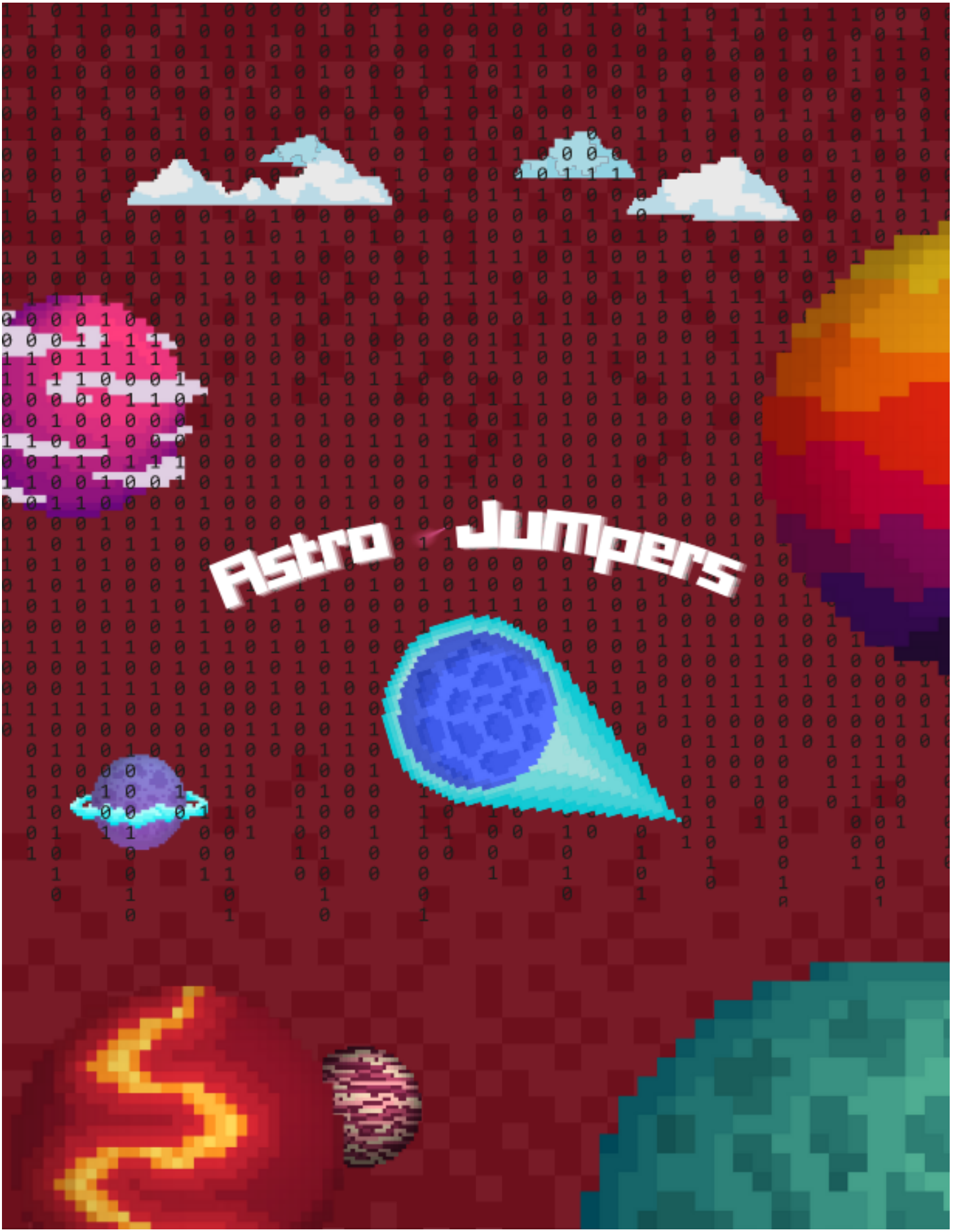

Astro Jumpers

ASTRO JUMPERS

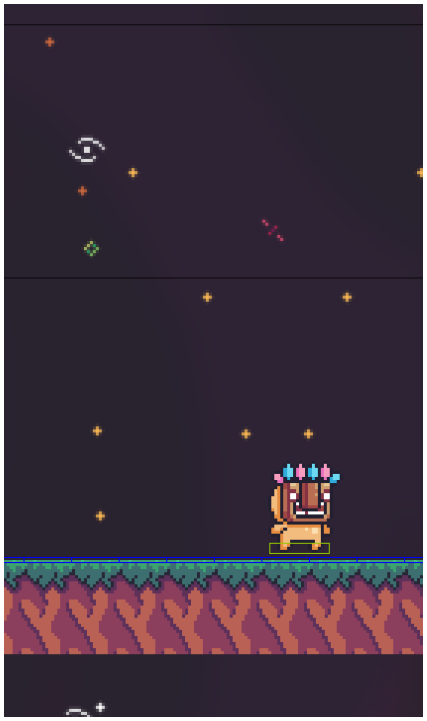
General Description

The game "Astro Jumpers" is an exciting infinity platform jumping video game that immerses you in a cosmic adventure full of action and challenges. In this game, you embark on an intergalactic journey as you control a brave character with the goal of discovering new horizons.

Key Points

- Console: PC
- Peripherals: Keyboard
- Genre: Arcade/Strategy
- Target Audience: 8+ years old
- Gameplay: Subway Surfers, Doodle Jump
- Style: 32-bit

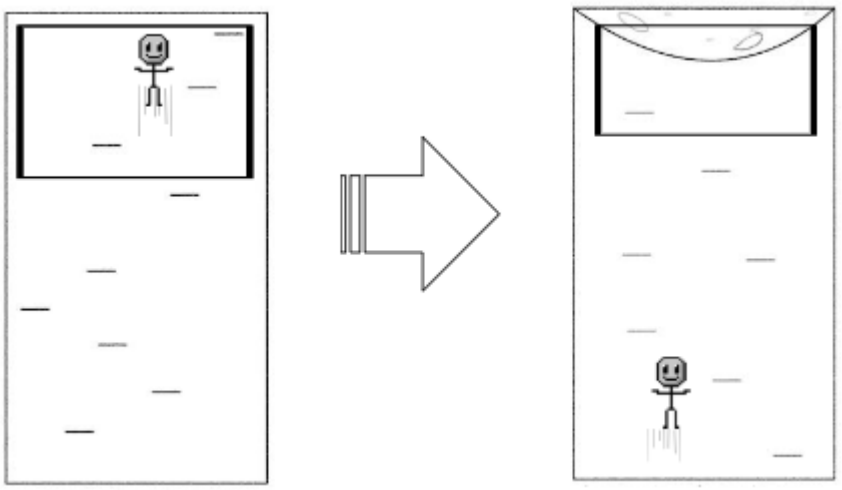
Game Mechanics



There are three types of dynamic scenarios with the vertical "Camera-Window" camera type, each one representing a different level. The first level is the easiest, and the last one is the most difficult. The size and spawn time of the platforms, as well as the player's gravity are some of the stage properties that are altered according to the specified level. Also, your limits are found in your parallels, but the sky has never counted as one, that's why your character will not be able to exit the stage from the sides, but will be able to travel vertically to infinity.

Within a specific level, your objective will be gravity. You will have to meticulously maneuver your character with respect to the randomly generated platforms with the objective of never falling from such a high place, because otherwise, you will die. Likewise, we generated a point system according to the maximum height reached, because conquering new horizons will always make you want to surpass yourself once again.

In this way, the user, when reaching the top of the camera, it will point higher, alluding to the fact that the player does not stop climbing. It should be noted that if the user falls to the bottom of the camera, he will lose, having to start from where the level began. In the same way, this will have limits on the side walls and on the upper part, because if the user runs into them, he will not be able to exceed that coordinate, nor will he lose, he will only run into a wall.



It's worth noting that your character's stamina is infinite, so you'll always find him jumping without the need to press any button. This can be both beneficial and detrimental at the same time, you are the only one who decides how to take this.

At the moment, you can only play alone, but the second version is being developed to allow the participation of several characters in the same scenario.

The only movements allowed to the main character are to move from left to right with the A and D keys, respectively. Only the BACKSPACE key is used to initialize the jumping state of that character, but this only needs to be pressed once in the game.

History and narrative

You wake up in a world where everything is new to you, the air, the stars, the galaxies around you are different. The air, the stars, the galaxies around you are different. What is the first thing you would do if you woke up in a totally unknown place? Explore it.

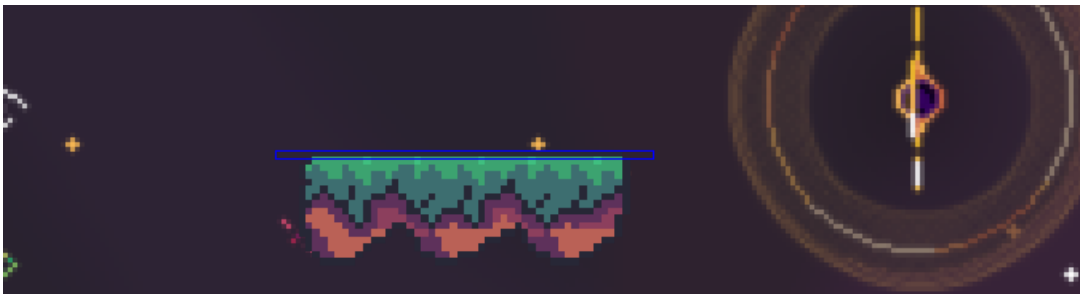
You decide to embark on your journey with simple jumps between platforms you find on your way, and you start to realize that you can jump and jump without getting tired, as you did before.

Existential questions arose, however he had no answer as to who he is, nor why he woke up in a place far from home, but he had an instinct that told him he should find out what lies beyond.

After several attempts and several falls, you still haven't reached the top.

This is the story of Shell, a person captured by scientists and subjected to experiments with the aim of turning her into a perfect deity of living flesh. Abandoned in an uninhabited world, her purpose is to test the hypotheses put forward by the researchers.

Will Shell be able to find the truth behind it all?



Characters and enemies

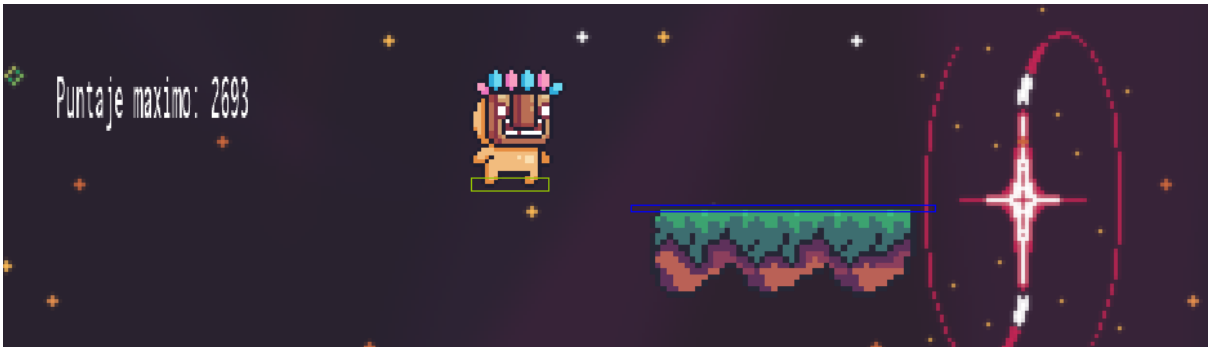
Shell is the main character of this videogame: an unidentified person with no knowledge of his surroundings subjected to various experiments by the scientists' guild on the planet where he previously lived.

The message of who Shell is can be easily seen through his costume, which is only a mask representing confusion in the face of self-discovery.

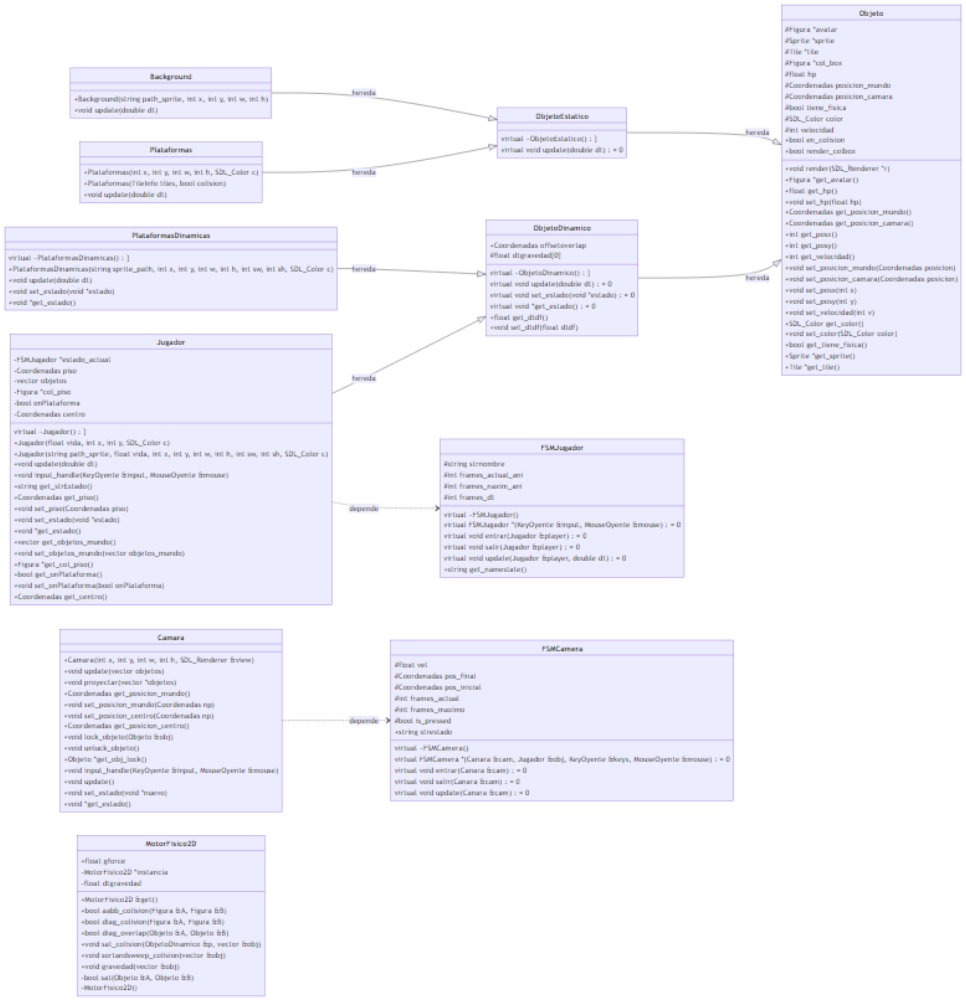
The same character does not understand that people with evil purposes were the ones who genetically modified him for their research, since he woke up without having any history of who he was, so Shell has no enemies. The only adversaries are gravity and the platforms, which are obstacles that prevent him from knowing the truth.

Objectives and progressions

The main objective of the game is to get the best score in each level. To do this, you will need to go as far as you can, since the points increase according to the height you reach. Within the player's story, it is contemplated that your objective is to find out what is beyond just platforms.



Design level



The UML diagram above is a graphical picture of the classes and some of the mechanics that happen in the game. Because our game is primarily on the Y axis, the camera does not need to move on the X axis, so it is locked on that axis, NOT allowing the player to move off the screen horizontally.

Inside the code there is an auxiliary class to SDLApp called SDLApp_AUX. As its name says, this class serves as auxiliary to SDLApp, allowing other functions (for example, MotorFisico2D) to access values without having to see everything that happens in SDLApp. This is important since we need that: MotorFisico2D, FSMPlayer and PlatformSpawn can read what difficulty the player is playing on.

It should be noted that within the SDLApp class there is an important function, and that is to show the player on the screen when his turn is over due to his death. It counts two seconds after the player's death, and then restarts the game by clearing the entire cache using the on_clean method. In between the two seconds, the screen displays no message other than "YOU ARE DEAD".

Below is an example of SDLApp_AUX in MotorFisico2D.

```
void MotorFisico2D::gravedad(std::vector<ObjetoDinamico *> objs) {
  for (auto &o : objs) {
    if (!o->get_tiene_fisica())
      continue;
    int cy = o->get_posy();
    // if(o->get_dtgf()!=0)
    if (o->get_dtgf() <
        (20 + ((SDLApp_AUX::get_nivel() - 1) *
               20))) // VELOCIDAD DE CAIDA !! //60=nivel3; 40; nivel2; 20=nivel1
      o->set_dtgf(o->get_dtgf() +
                  ((float)SDLApp_AUX::get_nivel() * 3 + 10) / 100);
    int y = cy + o->get_dtgf();
    o->set_posy(y);
  }
};
```

```
#include "sdlapp_aux.hpp"
#include "sdlapp.hpp"

int SDLApp_AUX::get_nivel() { return SDLApp::get().nivel; };

void SDLApp_AUX::set_nivel(int n) { SDLApp::get().nivel = n; };
```

When the player changes difficulty, the game is restarted and all the values depending on the difficulty are properly adjusted: gravity, player speed and platforms (you can change the difficulty by pressing P + [number]). Specifically, in level one, there is not so much gravity so that the user can move from platform to platform quietly, also the speed of the player is not so much. In level two, the gravity is increased a little, and at the same time the distance between platforms is reduced so that the user can move fairly, decreasing the size of the platforms significantly. In level 3, the platforms are too small, the gravity too large, and the platforms too close together, so that the user can quickly maneuver his movements. It should be noted that level one has 15 platforms that spawn randomly every second, and are removed after they are projected; level two has only 10 platforms; and level 3 has only 5.

```
if (KeyOyente::get().estaPresionado(SDL_SCANCODE_P)) {
  // printf("Pausa\n");
  if (KeyOyente::get().estaPresionado(SDL_SCANCODE_1)) {
    nivel = 1;
    KeyOyente::get().reiniciar();

    get().reiniciar();
  } else if (KeyOyente::get().estaPresionado(SDL_SCANCODE_2)) {
    nivel = 2;
    KeyOyente::get().reiniciar();
    get().reiniciar();
  } else if (KeyOyente::get().estaPresionado(SDL_SCANCODE_3)) {
    nivel = 3;
    KeyOyente::get().reiniciar();
    get().reiniciar();
  }
}
```

The player will be jumping platform after platform indefinitely until he falls and therefore "dies". When the player dies, a message is rendered warning the player that he died and the game restarts at the same level where he died.

The FSMCameras and Cameras classes were modified to allow the player to drag the camera upwards. The X axis is still available to the player, but it has a limit so that the player does not leave the camera.

Within the FSMJugador class, we modified the condition that when it enters the jumping state it does not stop being in the same state, because we want it to always be jumping our target.

The class PlatformSpawner Platforms helped us to form randomly on the x-axis (with a constant y-axis) on the map, all this depending on the level that the user entered. The values that are subject to change depending on the level are: the y-distance between spawned platforms; the maximum number of platforms contained within the map; the size of the platforms; as well as controlling the random x-spacing of the platforms.

The MotorPhysical2D class let us play with the gravity and the player's falloff, as well as shaping an optimal collision. Level 1 we let the gravity be 0.13f, level 2 with 0.16, and the third level with 0.19f; the more difficult, the more gravity.

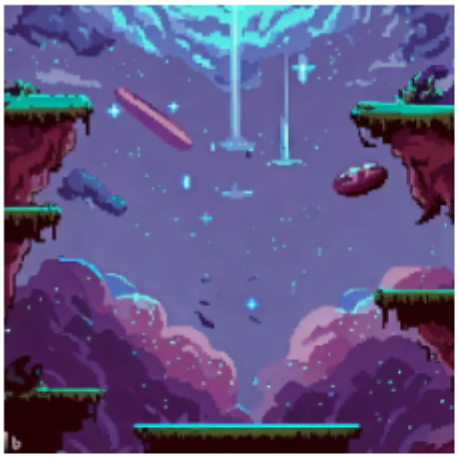
The class ObjetoDinamico is the class used for the creation of objects that are in constant movement or change, such as dynamic platforms.

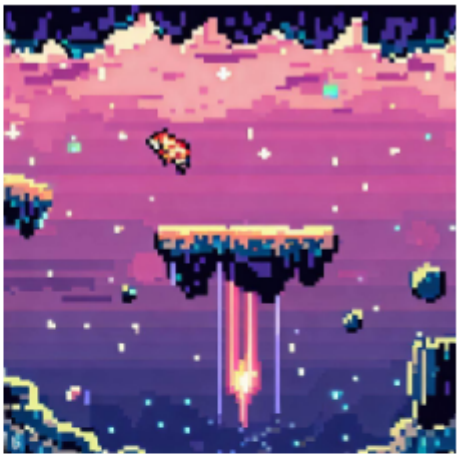
The StaticObject class is the class that was used for the creation of those objects that we needed to be stable, such as the row of tiles that is presented at the beginning, where the main player spawns.

Visual style and art

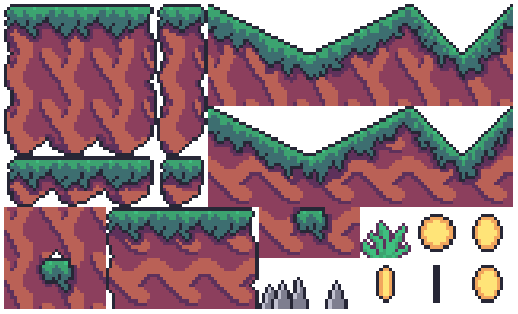
A style that has never gone out of fashion since its emergence is the 32-bit style. This style was predominantly used in this videogame because it wants to show the same intention as its emergence: never go out of fashion. Likewise, the combination of the galactic/space theme deepens the main theme, which is the protagonist's self-discovery. For the same reason, the vertical "Camera-Window" type of camera makes mention of always looking ahead no matter the obstacles.

Some of the inspirations used were the following:





The tiles used to give context to the scene were the following:



The ground block located in the upper left corner was used to generate the static objects of the map, which is the base where the user is at the moment of spawning.

The tile used to create the dynamic objects, i.e. the platforms that are spawned and un-spawned, was the following:

With this sprite, the player's jump could be observed in a dynamic way, an important state because the player will always be jumping:

This sprite was used to create the Idle mode of the character:

Finally, these are the backgrounds that were used dynamically to give more immersion to the player to belong in the intergalactic space: