Super NYAN GAME

SUPER NYANGAME GAME DESIGN DOCUMENT AND TECH SPEC DOCUMENT

INTRODUCTION

This document will serve to describe the inner and outer workings of our flash game: Super NyanGame. Everything will be covered in detail, from the gameplay to game flow, to the classes and objects themselves.

This game is an arcade, therefore, it just needs one level of increasing difficulty, but it’s adaptable enough to add more gameplay elements as enemies, power-ups, or even convert it into a level-based game without much problem.

TEAM AND WORK

Our (unfortunately with no real name) team is composed by Miguel Campins Fernandez and Óscar Gil Ferrer. We did not exactly split the work, but as the project went on, we focused on the parts that were best suited to both of us, and found out that we have a pretty good understanding.

GAME MECHANICS

CORE GAMEPLAY

Super NyanGame is a pretty simple but hard game. Easy to learn, hard to master. The objective is simple, help Nyan Cat collect as many stars as possible without dying.

The obstacles are alien invaders that want the poor cat dead. This is the point of freshness of our game. The enemies appear at random and have different capabilities! Therefore you will have to learn their patterns to make sure you don't crash into them while collecting stars.

GAME FLOW

The flow in this game is very direct once you start the game and get past the intro screen, you are launched directly into the fray. One may ask if it isn't necessary to explain the controls to the player, but as soon as the move the mouse a bit, they'll immediately learn the controls.

Once into the game, enemies and stars will start to appear on the right side, opposing the player. This lasts until the player has lost all of his hit points, then he will be directed to the Game Over screen, where his score will be shown. There, he can press the star, get back to the intro screen, and try again.

CHARACTERS

In this version we only have 3 characters.

Nyan Cat: The protagonist, who must collect as many stars as possible. His hit points are reflected in the height of the rainbow trail that he leaves behind. Numerically, he has 100 hit points.

Invader: The most basic enemy. His pattern is simple. He moves in a straight line going to the other side of the screen. He's a green invader from Space Invaders. Crashing into him makes you lose 10 hit points.

Destructor: A tricky enemy. His pattern is a sine wave, therefore, the player must know where to be knowing his trajectory so that he does not crash unknowingly. He's a red invader from Space Invaders. Crashing into him makes you lose 10 hit points. This is the “hard” enemy, so we do not allow them to spawn two or more in a row.

All 3 of them are different enough so that they don’t need any more description.

GAME PLAY ELEMENTS

Aside from the player and the enemies there is only one more game play element: the stars.

The stars have the same pattern as the Invaders, they go from right to left in a straight line, but with one peculiarity. They come in groups from one to five. On impact with the cat, each of them gives him 1 hit point back to a maximum of 100 hit points.

Collected stars are the player score, however every time a star is collected we accelerate the flow of the game. In this way, difficulty increases as long as player progress through the game.

GAME PHYSICS AND STATISTICS

Given that our game is based around flying through space, we had to give the impression that that is what is really happening. Using a moving background and using a flowing rainbow trail animation, we can make a “realistic” scenario where the cat is flying and generating the trail instead of both of them fixed images, as the cat cannot move in the X axis.

To add to the illusion, we made the obstacles go from right to left, therefore, they seem that they are really unmovable when it is really the obstacles that are moving instead of the cat.

USER INTERFACE

FLOWCHART

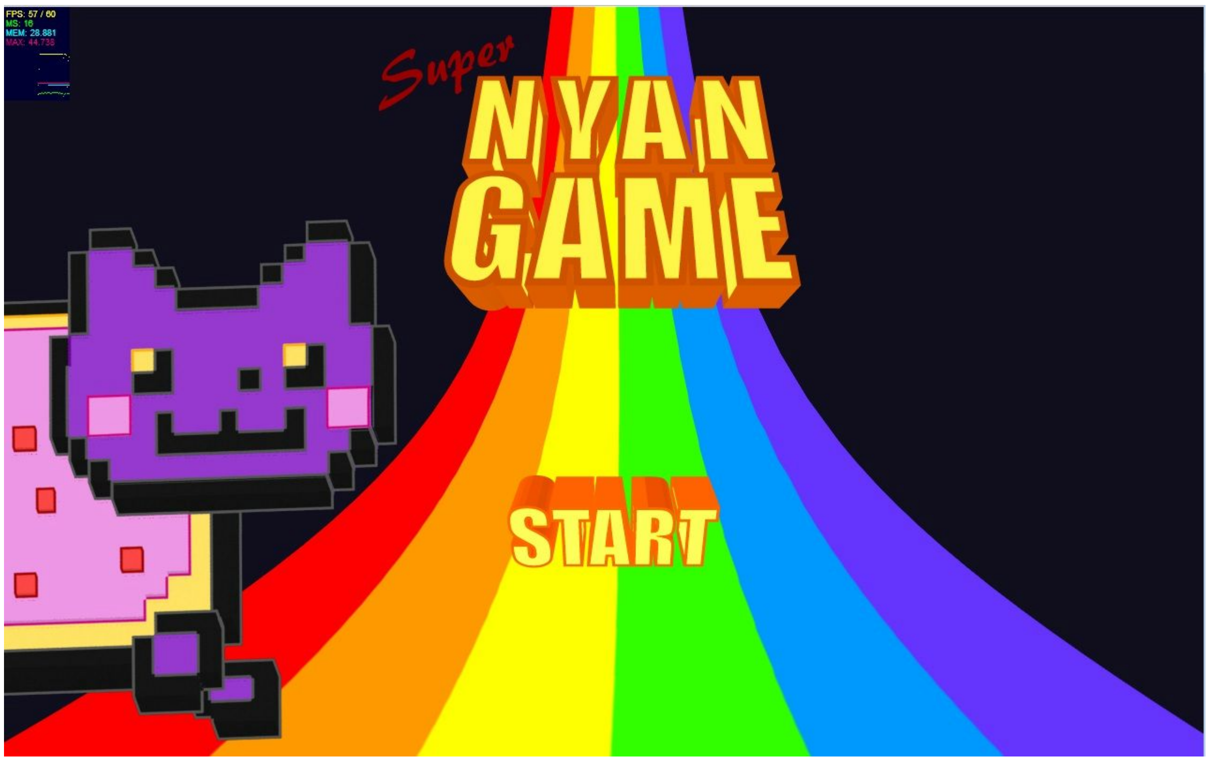
Intro Screen -> Game Screen -> Game Over Screen -> Intro Screen (loop)

FUNCTIONAL REQUIREMENTS

-Intro screen:
In the intro screen one can see the title of the game and start it by clicking or tapping into the START button.

-Game Over screen:
This screen shows the final score of the actual session and if the player presses the star button, it returns he/she to the intro screen.

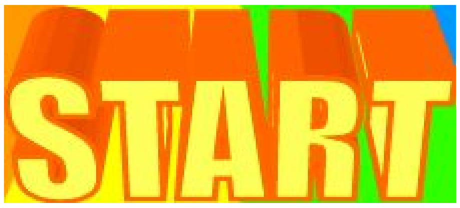
MOCKUPS



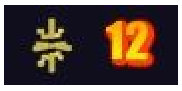
```
FPS: 57 / 60
MS: 16
MEM: 28.881
MAX: 44.738
```

Intro screen

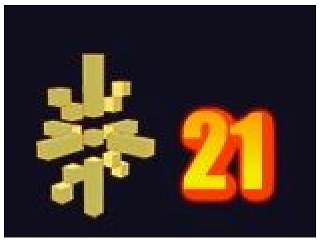
GUI OBJECTS



Start Button



Score text



Final score text and star button

ART AND VIDEO

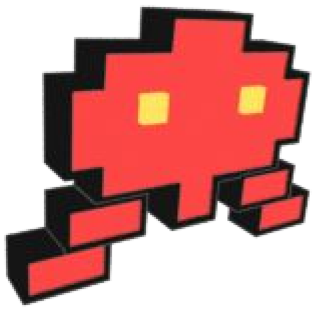
OVERALL GOALS

The objective of using this art style was to try how would 3D voxel art style would look like in 2D.

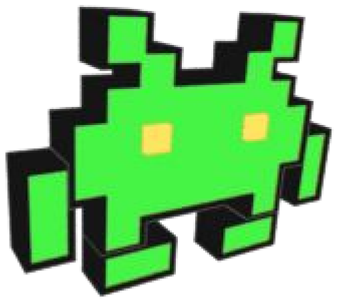
The result has been a kind of “out-of-the-screen” experience, as it looks like a 3D game even when it is not.

2D ART AND ANIMATION

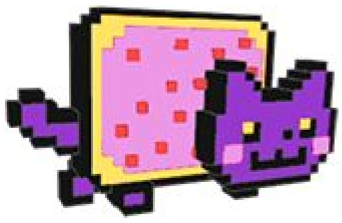
Gameplay elements

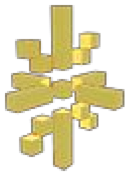


Destructor



Invader


Nyan Cat



Star

TECHNICAL SPECIFICATIONS DOCUMENT

GAME MECHANICS

-Avatar

Your avatar in this game is the Nyan Cat. For those who are not familiar with it, Nyan Cat appeared as a meme on youtube, where the aforementioned cat flew through space with a loop of a song whose lyrics were simply “nyan”.

His only ability is to move up and down.

-Controls

The game is designed with touch controls in mind, so you can move the cat sliding the finger up and down or, in the PC version, you can simply move the mouse and the cat will follow its movements.

-Goal

As said before, you goal is to collect as many stars as possible while avoiding the invaders and destructors. Collecting stars will also heal the Nyan Cat if you got hurt.

The game lacks any story whatsoever right now, so it’s not possible to justify it, but we associated space and stars with it since in his original video the cat is flying through a blue background filled with stars.

-Death

Cat spins with a sad face whenever you are hit and the rainbow trail gets smaller. When you are out of hit points, it is Game Over.

-Scenary elements

The core elements that are shown on screen are the UI, the invaders, the destructors, stars and player avatar as long as their current hitpoints in the form of a rainbow. There is also a background of stars that move against the cat depending on their scale, creating a feeling of depth.

-Interaction

The cat can interact with the different elements by crashing into them. Crashing into a star will give the cat 1 hitpoint and crashing into an invader or a destructor will substract 20 hitpoints and will reset its velocity. If the cat manages to fly unbeaten, he will start to accelerate progressively.

-Tools/Upgrades

As of now, gameplay is very simple, but thanks to the mechanics of the spawner and the classes, it is very adaptable to change the patterns of the obstacles, create new obstacles, or even create power-ups for the cat.

·Platform and OS

The game runs on Flash Player, therefore, in any browser up to date.

·External code

We made use of the following libraries:

Starling: An actionscript framework that focuses on web game design.

Hi-ReS-Stats: A little software to monitor the game's performance while it is running. We used it for testing purposes despite it won't be shown on the release version.

TweenLite: A library focused on making fluid animations.

·Control loop

Main.as is used mainly to initialize starling and pass its control to Game.as, who has the responsibility of initializing each of the screens when a navigation event is dispatched to it: welcome, to access the intro screen, play, to access the game screen, and over, to access the game over screen.

·Game object data

Each object has enough data to function by itself without really depending on any other class, and that is what makes this game so adaptable.

MAIN

The main class merely initializes the starling framework and gives control to GAME.

GAME

The core of the game. Where every other screen is initialized when it is needed.

ASSETS

The class that stores all the assets in the game.

CAT

The class of the main character. It includes the drawing of the character’s art.

OBSTACLE

The class of the 3 types of obstacle. It's builder works in a way that when it receives a number, it actually creates that type of obstacle. By uniting all of the obstacles in one class instead of making one for each, spawning mechanics are a lot easier and can make possible to expand the number of obstacle types without the hassle of having to create a new class.

NAVIGATION EVENT

This class holds the conductor of the screen changes.

WELCOME/INGAME/GAMEOVER

This are the three classes that compose all three screens that appear in the game.

Welcome is a pretty basic class that puts all the elements that compose the intro screen and a button to start the game.

On the other hand, InGame is where the meat of the game occurs. Here, we have to work the initial position of the player character, and the obstacle spawner. The obstacle spawner works on a percentage basis. 60% of the time, an invader is spawned. 30% of the time a star is spawned, and a 10% of the time, a destructor is spawned.

Another detail is that picking a star, will decrease the delay between spawns, making the game more difficult.

And lastly, the simplest screen, GameOver. It just shows the final score and a button to return to the intro screen.

DATA FLOW

The data flow is mostly self-contained, with each class having its own data, and the only data that is transferred between screens being the final score, from InGame to GameOver.

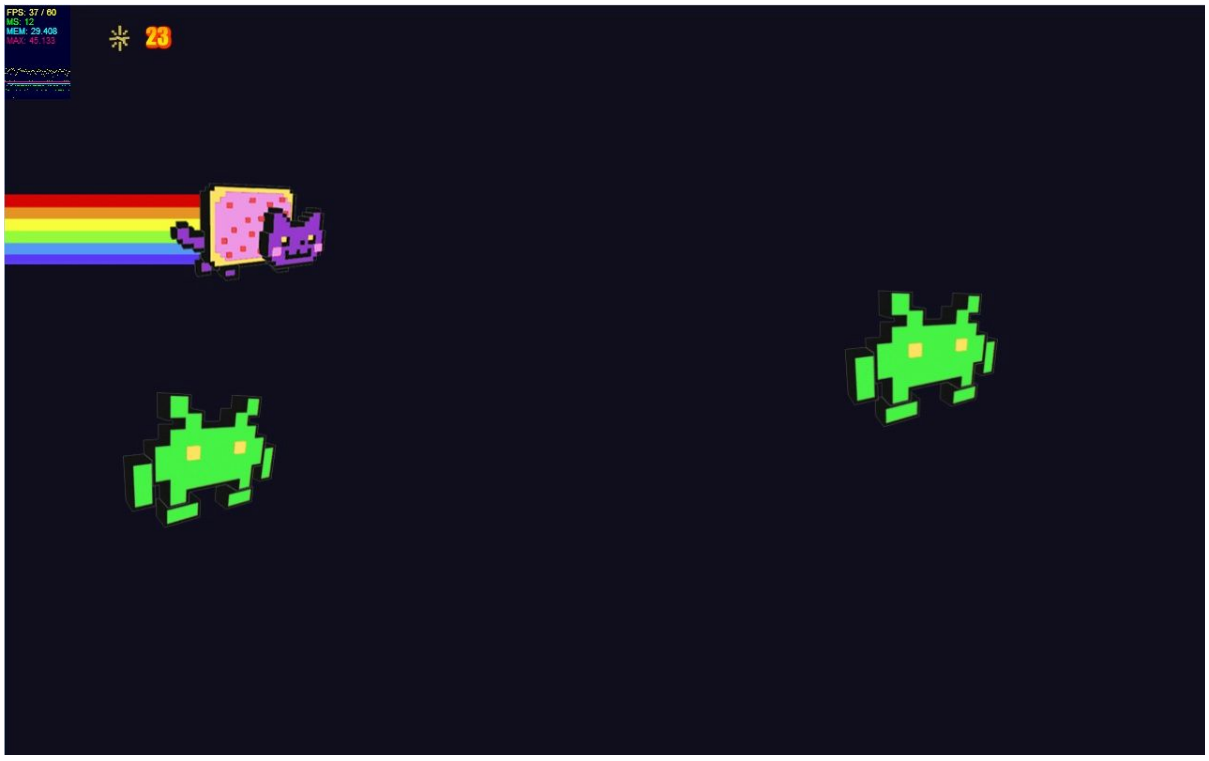
USER INTERFACE

The user interface consists of two buttons (the start one, and the retry one), and the control of the cat.

At first, the cat cannot be controlled until it arrives to his X axis final point, where it can finally be controlled, by simply moving the mouse, or sliding your finger.

MAIN PLAY SCREEN

Here is a screenshot of the main play screen:



Here we can we the cat and two invaders, and an actual score of 23 stars collected. In this build, the background is not yet implemented.

ART AND VIDEO

Every character art was created on a 3D editor software and has been pre-rendered most like at the end of 16-bit age, when games started to widely explore the 3D environments. All art has been implemented via embedding sources.

GAME BUDGET

Here is a breakdown of the game budget:

Programming	3.000€/month	2 employees
Art and animation	1.000€/month	1 employee
Level Design and Testing	1.000€/month	2 employees
TOTAL	5.000€/month	

Assuming a total of 6 months of development, the total cost of the game would be 30.000€