Kadick's Resolution: Using procedural design for the development of a Web Game

Final Degree Work
Bachelor's Degree in
Video Game Design and Development
Universitat Jaume I

July 1, 2019

Author: Manuel Gavilán Ortiz

Supervisor: Juan Carlos Amengual

ACKNOWLEDGMENTS

I would like to thank my Final Degree Work supervisor, Juan Carlos Amengual, for his guidance and advices in order to make this project become real.

To my parents, for allowing me to follow my dreams.

To María Saborit Ribelles, as she is the one who shared this path by my side being the best partner a man can wish for.

Lastly, I would like to thank my classmates and friends, for their friendship and the priceless moments we shared and won't be forgotten.

ABSTRACT

This document presents the end-of-degree project for the Bachelor's Degree in Design and Development of Video Games. The main idea for this project is to develop a videogame for internet browsers.

Kadick's Resolution is a Roguelike game imbued with cyberpunk style in 2D top perspective. Its scenario is generated using automatic generation through procedural design. This game uses Javascript as the main programming language including the Phaser framework to improve and ease the development process.

This game is fully localized in English, Spanish and Valencian.

INDEX

FIGURES

1. INTRODUCTION

In this section it is detailed the motivation behind this project and the objectives that it is expected to fulfill. There is also a brief explanation about how the procedural design is used nowadays to improve replayability.

1.1. WORK MOTIVATION

This project is the consummation of two different degrees: Technical Engineering in Telecommunications and Design and Development of Video Games. The author desires to combine his experiences during both his academic and professional career.

As a technical engineer in telecommunications, the author has found love and passion in developing tools for web applications and pages. For two years, the knowledge acquired in web oriented languages has improved the author's perception of entertainment and easy-to-use environments. Thus, this project had to be implemented into a web site.

As a game designer, the author found it of utmost importance to bring his experiences -not just as a professional in the field but as a passionate player as well- into this project. Being a player for more than twenty years brings up an extensive variety of patterns for a game to be developed. However, it had to be web-friendly. This means that many limitations had to be taken into account: lower level programming, web performance, memory leakage and rendering limitations. Fortunately, the author loves a great challenge.

1.2. MAIN GOALS

- Design and create scenarios, characters and enemies taking into account the cyberpunk reference.
- Design and develop animations for the main character, enemies, bosses and power-ups.
- Develop and integrate a power-up system that improves the final experience while playing the game.
- Develop a procedural system that generates a random scenario for each stage.
- Develop a random enemy spawn system.
- Develop an easy to understand visual interface.
- Create and manage the main data structure in order for it to be scalable and also to perform localisation.
- Perform localisation in English and Spanish and manage an i18n file so new languages can be added in the future.
- Develop a responsive movement set which implies the usage of keyboard and mouse.

1.3. STATE OF THE ART IN PROCEDURAL DESIGN

Video games are widely considered a form of art. Their complexity is rooted in merging a great plot into an interactive media that allows the player to become an actor. This means that the user is no longer an outside observator but rather the main character that manages the flow of the narrative.

For many years there has always been a limit to this kind of experience. To be more precise, the main character had to perform on an scenario encapsulated by predefined boundaries, with a limited amount of enemies set into predefined locations. This eases the job of the main designer but results in lowering the replayability and the sense of realism as the world and reality itself is relatively aleatory. Deterministic values makes a game predictable. This fact does not mean a game is necessarily bad at all if it is not randomly generated. It means that the player will always know what lies ahead each time he replays the game.

Procedural generation is a method that implies a computer generating pseudo-random content based in programmed basics. It means that every time the player starts a new game there might be changes in the layout, loot, enemy spawn, quantity and disposition. Thus, it is impossible for the player to know what awaits on his path.

There is a vast catalogue of games that implement this procedure into their code.

The *Diablo* [1] saga by *Blizzard* [2] performs a pseudo-random generation of its scenario. Main locations as cities and emplacements are constants. However, stepping into the wilderness or entering a secret basement or a dungeon means facing the unknown. The distribution of every cave and dungeon is different. The same goes for enemies spawn and disposition. Elite warriors are also randomly generated and their special abilities are also random and displayed on the life gauge as it is seen on Fig. 1.

Other games such as *Minecraft* [3] and *Terraria* [4] implement a world procedural generation. This means that every new game consists on a new world generated from the beginning following a set of rules and specified biome configurations: forest, desert, ravines or oceans are some examples. These rules also determine enemy spawns and NPC (Non Playable Characters) placements. As seen on Fig. 2, the game displays a loading screen while the world is being built.

Fig. 1. *Fight against an elite enemy Diablo III. Blizzard.*

Fig. 2. *Generating message in Minecraft and Terraria*

Finally, there is one genre that has become famous over the years: The roguelike. This genre can be defined by these characteristics.

- They are mostly single player games.
- The dungeons and scenarios are randomly generated.
- Playability is of utmost importance, surpassing graphic design.
- Simple plot.

The perfect example for this genre would be The binding of Isaac [5], which is the game this project is more based in. This game performs random values for almost everything. The disposition of the cells that conform the dungeon is randomly generated. The size and form of each cell is also random, as it is the enemy spawn, the power ups dropped by enemies and even the secret rooms. Fig. 4 shows an example of a randomly generated room.

Fig. 3. *Random room. The Binding of Isaac.*

The simple yet entertaining playability of *The binding of Isaac* along with the main idea of generating a random scenario using procedural design inspired this project.

1.4. TOOLS OF DEVELOPMENT

- Programming: Visual Studio Code.
- Art and animation: Photoshop
- Browsers: Google Chrome, Mozilla Firefox, Microsoft Edge.
- JS Framework: Phaser.

1.5. WHY PHASER 3?

Phaser 3 is a JavaScript high level video game oriented framework. It is the newest version of the Phaser framework and allows ES6 scripting.

There are many reasons for choosing Phaser. First of all, it is really easy to get started with. It has a really easy to understand API and a vast number of tutorials and examples. There is a really helpful community that shares valuable knowledge and experiences in order to grow as a game designer. Furthermore, Phaser supports third party libraries and plugins.

1.6. PLOT AND SETTING

Kadick's Resolution

Kadick is an android designed and developed by Asimov Industries. As one of the highest ranks in the street guardian group, it must protect the citizen and destroy the villain. During its last update a part of its operative system got corrupted. Thus, the software that inhibits the functionality of its AI to question its actions disabled itself. Tormented by its own nature, disregarding what its core tells it to do, its motive is now clear: To get to Asimov, its creator, and get some answers. This, however, will not be easy.

'*Why am I to destroy the ones I was created to protect? How do I know what I am doing is right and not just what I was programmed to do?*' — Kadick.

Characters

Kadick: The main character. An android designed to maintain order in the streets. Its AI fails causing him to consider its own actions.

Asimov: CEO of Asimov Industries. Self-declared philanthropist. Developed the Street Guardian Force to maintain the order where the government could not. His motto is: "Hit first, hit fast, hit hard and, by all means, be the last that hits".

1.7. RELATED SUBJECTS

VJ1217: Web Game design and development is the main related subject. This subject currently uses the Phaser 2 framework. This project pretends to take a step further.

VJ1237: Video game localisation is indirectly related as one the main objectives is to fully localize the game in order to reach a larger audience.

VJ1204: Artistic expression and **VJ1223: Video game art** form the main basis for the art style.

VJ1224: Software engineering is related to the usage of git as the project version control. It has also been helpful in order to divide the tasks in smaller ones and to develop efficiently.

VJ1222: Video game conceptual design is directly related to the design and development of the scenario and its characteristics.

VJ1215: Algorithms and data structures. The knowledge acquired in this subject has been used to develop the procedural generation of the scenario and the game data structure.

This page has been left intentionally in blank.

2. PLANNING

The initial planning for this project and its follow-up evolution will be very clearly explained in this section.

2.1. INITIAL PLANNING

The planning for this project had to be very precise as the schedule had to allow combining professional work with the final degree project development. This is the first approximation established in order for the project to be completed in 300 hours:

Task	Hours	Main Aspect
Design first sketches of main character, power ups and enemies	10	Visual
Draw and animate visual elements	20	Visual
Define and write text elements: script, UI, buttons...	10	Visual
Define and design main menu	5	Visual / Program
Design and integrate UI	5	Visual / Program
Define, create and integrate the procedural system	30	Program
Design and integrate random enemy spawn	5	Program
Design and integrate player's move set	10	Program
Design and integrate POWER-UPS	15	Program
Design and integrate player's attack system	15	Program
Design and integrate UI variables and their reactions to the gameplay	10	Program
Design and integrate enemies attack system	50	Program
Design and integrate boss attack system	40	Program
Game audio	10	Program
Writing Final Memory	50	Presentation
Preparing Final Presentation	15	Presentation
Final Estimated Time	300	

Once the project started, this approximation had to be taken into consideration and there had to be established an order of progression.

First, the development was to be focused on programming and developing the basic functionality. This implied that the art design would be limited to just creating the necessary sprites for each functionality and level. Animation and final art design and details were decided to be integrated in later stages in development.

Development was divided into stages in order to progress consistently. These stages were:

1. **Stage 1**: General Aspects. Basic gameplay elements were to be developed during this stage: movement, attack, disposition of player and enemies, scenario generation.
2. **Stage 2**: Corresponding to Level 1.
3. **Stage 3**: Corresponding to Level 2.
4. **Stage 4**: Corresponding to Level 3.
5. **Stage 5**: Corresponding to textual content and cinematics.
6. **Stage 6**: Corresponding to art and animation implementation.

2.2. FINAL PLANNING

The programming part was developed by dividing the whole project into stages and developing each stage in order. Fig. 4 displays the distribution of stages in time.

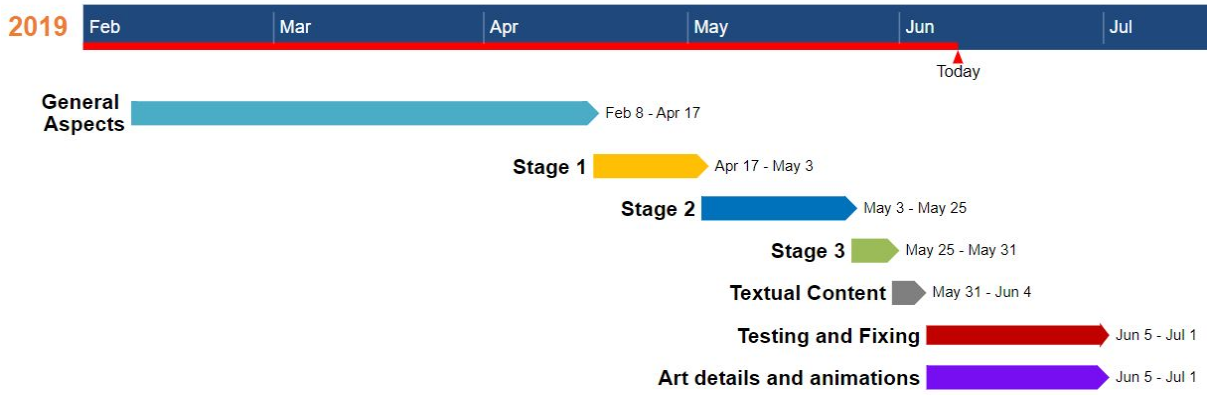


Fig. 4. *Kadick's Resolution development timeline*

General aspects include the basic functionality and the core of the game. This stage represents the data structure, character movement and data, difficulty settings, scenario procedural generation, prop disposition on scenes, basic user interface and room functionality.

General aspects were divided into small tasks. This table represents the amount of hours necessary to complete each task.

Task	Estimated	Final
Design first sketches of main character, power ups and enemies	10	10
Design and integrate UI	5	5
Define, create and integrate the procedural system	30	20
Design and integrate random enemy spawn	5	10
Design and integrate player's move set	10	5
Design and integrate POWER-UPS	15	15
Design and integrate player's attack system	15	10
Design and integrate UI variables and their reactions to the gameplay	10	15
Final Estimated Time	100	90

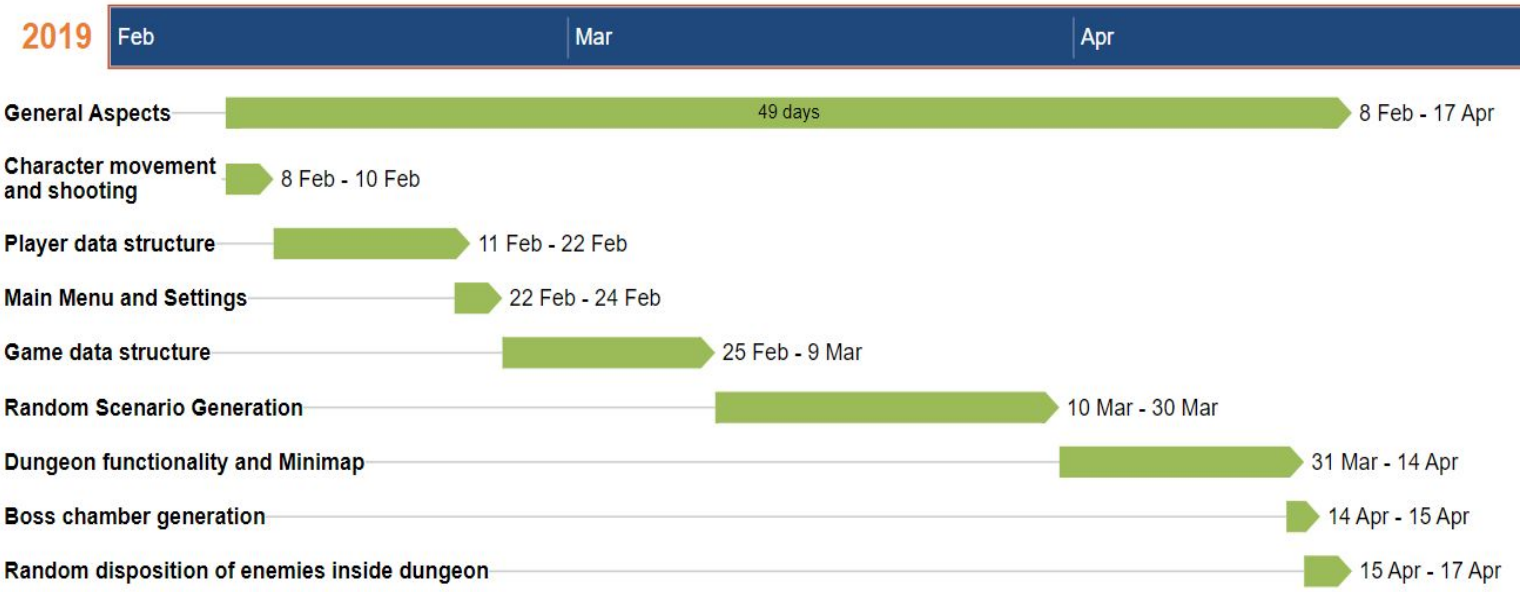


Fig. 5. *General Aspects Timeline*

The random scenario generation took great part of the development due to its complexity. Fig. 5 displays the distribution for each stage in time.

Dungeon functionality and the minimap represent the disposition and functionality of the elements at the scene. Door position, door types, the basic movement between different cells and the basic UI. The boss chamber is generated after the scenario has finished its disposition. Lastly the enemy spawn and disposition was implemented.

Fig. 6. Stage 1 Timeline

The stage 1 was decided to be easy in order for the player to get used to the gameplay. The Scancatcher was the first enemy made. Its artificial intelligence is very simple as its only job is to follow Kadick and hit him directly. The Jolt is the first enemy that will shoot the player at sight and will display a force field after being shot.

The bossfight was decided to perform three different attack patterns and was harder to develop as seen on Fig. 6.

Fig. 7. Stage 2 Timeline

The stage 2 introduced two new enemies. The coulomb is a mechanical bull that tackles the player until it reaches any kind of wall or obstacle. The trashbot is a neutral enemy that will not hurt the player directly by approaching him. Instead, he leaks oil on its movement and after some time it sets the oil on fire. Thus, the player has to avoid getting on the oil trail while dodging hits and bullets. As seen on Fig. 7, the trashbot was harder to create as the fire system was harder to recreate.

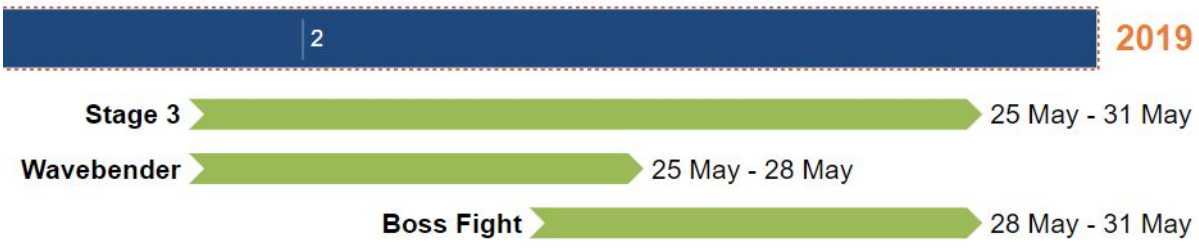


Fig. 8. Stage 3 Timeline

Stage 3 introduced the last minion: the Wavebender. Its name comes from its capability to generate pulses of energy that will hurt the player on contact. The last bossfight was the last enemy to implement. It was easy to implement its movement, not so the attack pattern.

Task	Estimated	Final
Design and integrate enemies attack system	50	60
Design and integrate boss attack system	40	40
Final Estimated Time	90	100

Developing the textual content was a task that involved designing the system to use the localization file in order to scalate it in future releases and make it easy to integrate through the game. In this task also comes designing the final aspect of the main menu, which included both the settings screen and the control screen

Task	Estimated	Final
Draw and animate visual elements	20	20
Define and write text elements: script, UI, buttons...	10	10
Define and design main menu	5	5
Final Estimated Time	35	35

Lastly the process of designing the animations and art details will run during the stage of testing and fixing as they are not mutually excluding.

This page has been left intentionally in blank.

3. DEVELOPMENT

3.1. PROCEDURAL GENERATION OF SCENARIO

The procedural generation of the scenario forms the main core of the application and has to fulfill these requirements:

1. The rooms must be disposed randomly both horizontally and vertically.
2. There has to be exactly 3 rooms where keys will appear once the enemies are beaten.
3. The start room cannot be a key room.
4. The boss chamber cannot be a key room.
5. There has to be a boss room.

This algorithm is applied every time a stage is generated. The scenario is represented inside a 10 x 10 matrix of cells and only those determined as playable will be rendered.

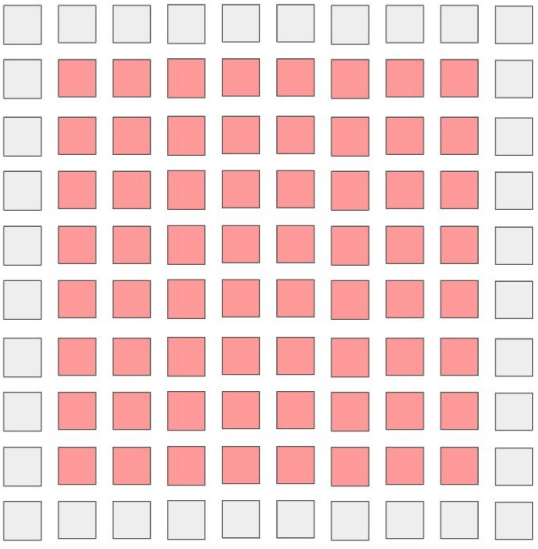


***Fig. 9**. 10x10 matrix of cells available*

The process of generating the scenario starts by selecting randomly a start room in the central 8x8 matrix of nodes as seen on Fig. 9. Therefore, the start room will always have available all four rooms vertically and horizontally.

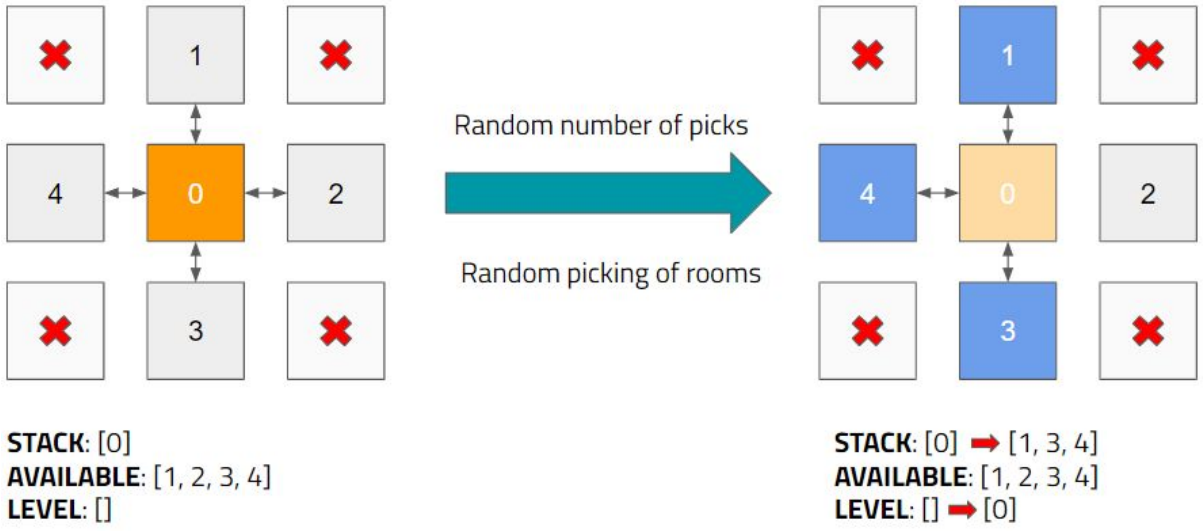


Fig. 10. *Random picking of available rooms*

Once the start room has been set, it is pushed inside a node stack. After that, the recursive method begins. This method takes out the first element inside the stack and checks its adjacent rooms in four directions: top, bottom, left and right. As seen on Fig. 10, the rooms at the corners are not reachable. Then it checks whether they have already been visited by the algorithm or not. If a room has not been visited, it is available. Once the available rooms have been evaluated, the algorithm sets a random number of rooms between 1 and the number of available rooms and picks them randomly. Those nodes are then pushed in order inside the stack and this process repeats itself until there have been added a minimum amount of rooms previously established.

Every time a room is taken out of the level stack is pushed inside a level array that conforms the scenario. As soon as the recursive process finishes, this array contains all the rooms that will conform the stage. It is now time to set the key rooms.

Fig. 11. *Array of nodes that conform the scenario*

To pick the rooms that will generate a key the scenario array is now used. This procedure selects three random rooms between the second position of the array and the last one. This is easy to do as the start room will always be the first element and cannot be a key room. See Fig. 11. Once the key rooms have been selected, what remains is to generate the boss chamber.

In order to create the boss chamber, we pick the last element of the array and check the available adjacent rooms. The first available room found will become the boss chamber. Fig. 12 displays an example of this procedure.

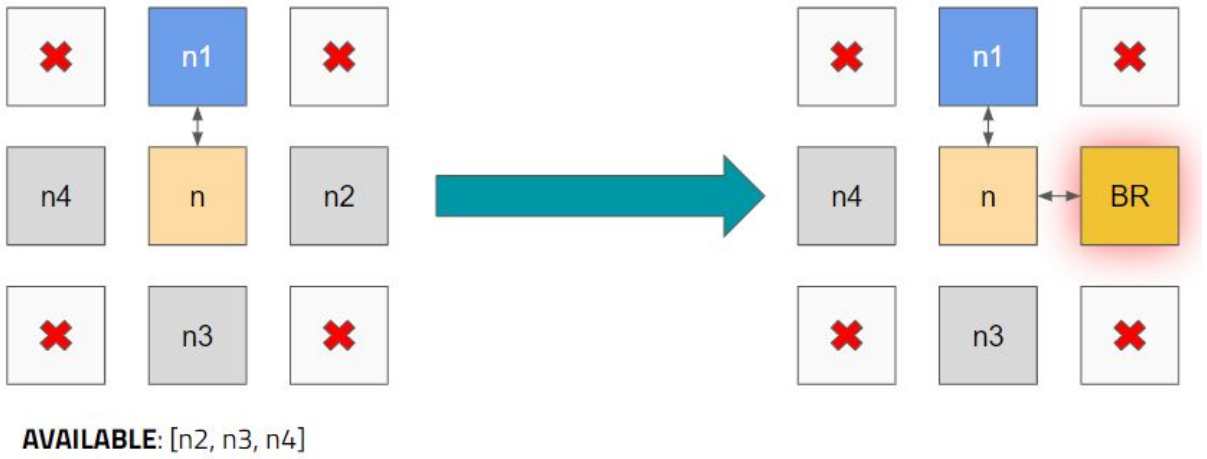


Fig. 12. *Creating the boss chamber*

This is the last step of the scenario generation. This is the final aspect of a random scenario. Orange squares represent the doors between rooms. Fig. 13 shows an example of the map generated using this procedure.

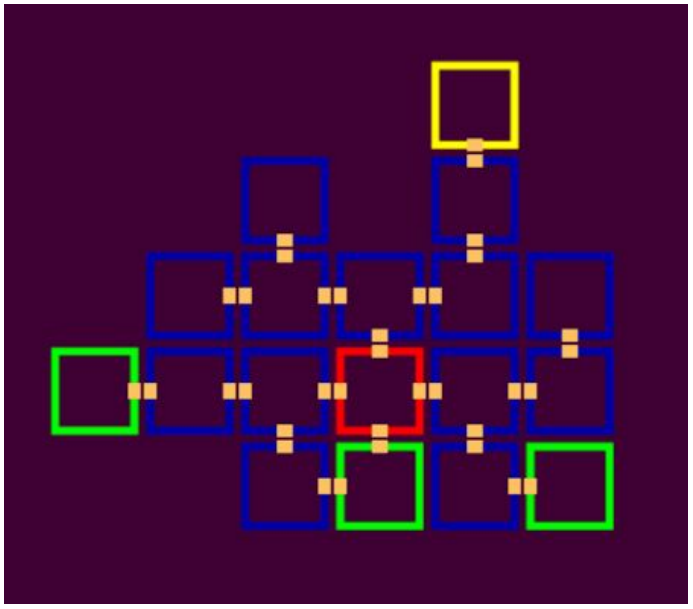


Fig. 13. *Randomly generated map.*

Red room: Start room.
Blue rooms: Scenario hostile rooms.
Green rooms: Key rooms.
Golden room: Boss chamber.

In order to fully understand how the scenario is managed, this is the information inside every room node:

Value	Type	
top	Boolean	Room has a room available to the top
right	Boolean	Room has a room available to the right
bottom	Boolean	Room has a room available to the bottom
left	Boolean	Room has a room available to the left
visited	Boolean	Room has been visited by the generation algorithm
isStart	Boolean	Room is the start room
isKey	Boolean	Room will generate key after beating the enemies inside
isBoss	Boolean	Room is the boss chamber
isClear	Boolean	Player has beaten all enemies in this room
keyIsTaken	Boolean	Player has taken the key from this room
distribution	Number	Obstacles distribution ID
whereIsBoss	String	Direction of the boss room. Empty if it is not adjacent.

3.2. ENEMY RANDOM DISPOSITION

The number of enemies displayed on scene depends entirely on the difficulty settings. The harder the game is, the more enemies will appear on scene. Their disposition depends on the position of the player as he enters the room. This algorithm takes Kadick's position and delimits an area that covers the part of the room where the player is not at. As seen on Fig. 14, the red area is an example of this available area. After that, there is a range of enemies of each time that will appear on scene. This range is determined by the difficulty setting. Once created, they are placed randomly inside that delimited area in order for the player to have time to react and act accordingly.

Fig. 14. Limited areas for enemy spawn

3.3. SCENE RANDOM DISTRIBUTION

The scene distribution of obstacles is also random. These obstacles limit the movement of the player and enemies. They also serve as covers from enemy bullets. There are several types of distribution and they all are referred to relative positions on the scene. These elements are stored inside a JSON file and loaded each time a new room is started. Thus, the distribution and size of the room adapts itself to the screen resolution as seen on Fig. 15. If the window were to be resized, the room would reload. This causes the distribution to change and the enemies to respawn.

Fig. 15. *Scene random distribution example. Displayed obstacles.*

3.4. PLAYER MAIN GAMEPLAY

The movement of the player is limited to eight directions as it follows the same moveset of a classic arcade game. The player will always aim at the cursor and shoot pointing at its direction. The laser shot by Kadick has unlimited range and will only disappear once it touches an obstacle or one of the walls.

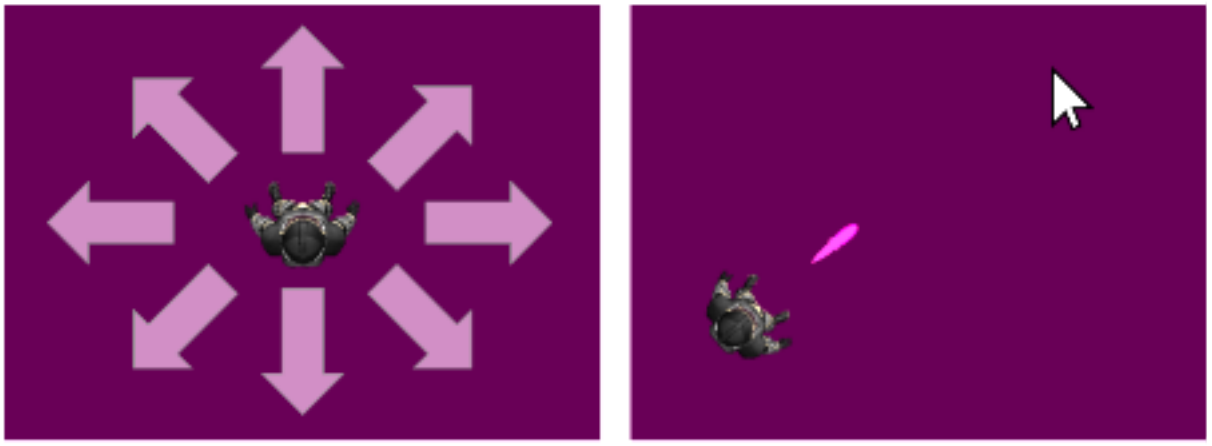


Fig. 16. *Player movement and shooting*

The main controls are set to the WASD keys and Kadick will shoot by using the left click. The control screen is available in the main menu for the user to check at any time. Fig. 17 shows a part of the controls screen, seen on Fig. 18.

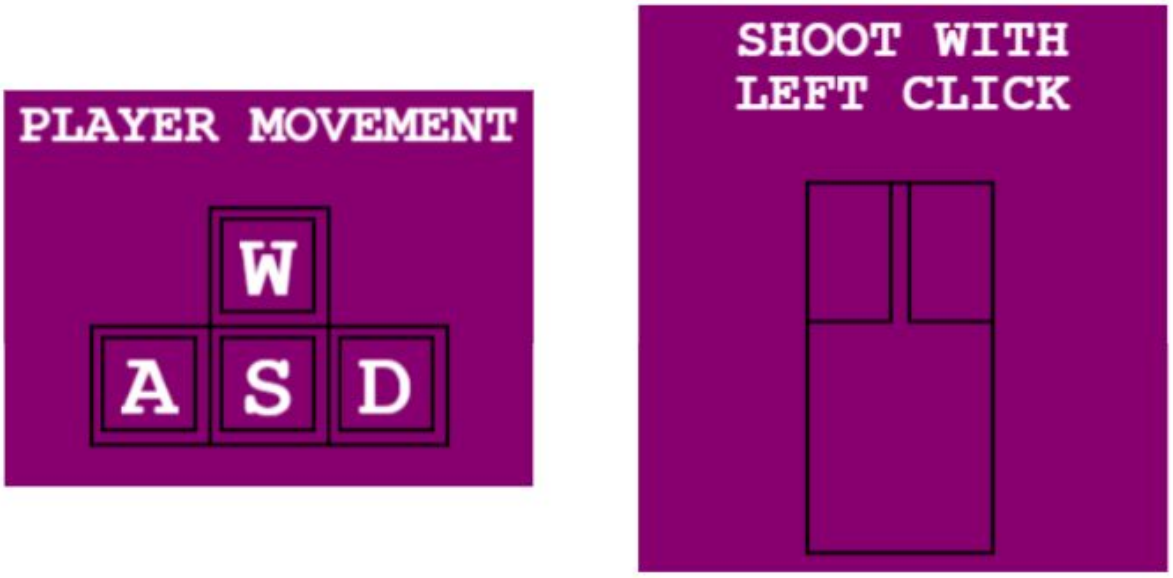


Fig. 17. Player Controls

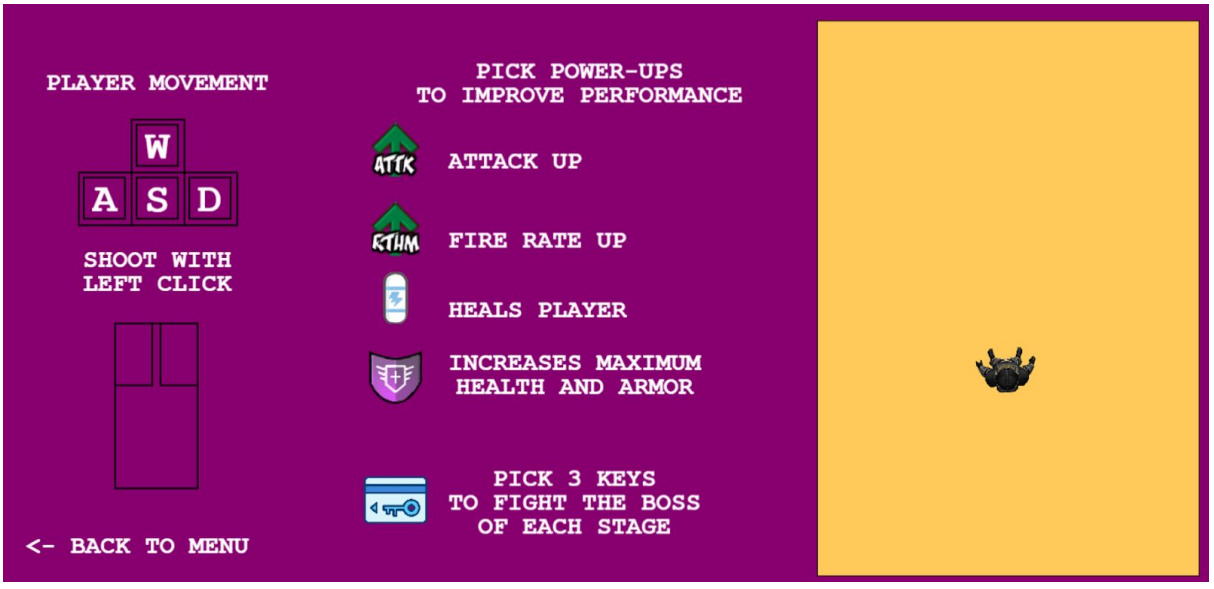


Fig. 18. Controls Screen

3.5. ENEMIES

There are five types of minion enemies in Kadick's Resolution. Each one of them has a different attack system and has an specific function as the player gains power.

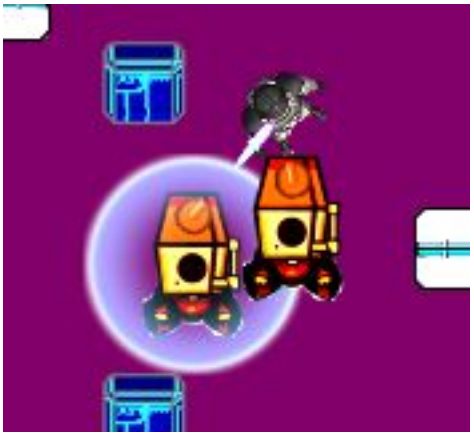
3.5.1. SCANCATCHERS

Health Points: 60
Damage: 20

Attack System: *Scancatchers* follow the player around the scene and hit him directly

Main objective: The player learns to move around the scene while shooting

3.5.2. JOLTS



Health Points: 80
Damage: 30

Attack System: *Shoots the player*

Main objective: The player learns to dodge bullets while moving.

Jolts display a force field when hit in order to protect themselves from the player's lasers

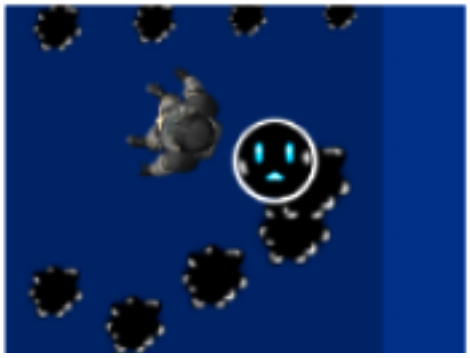
3.5.3. COULOMBS

Health Points: 200
Damage: 35

Attack System: Rotates towards the player. Then tackles him until it hits an obstacle.

Main objective: The player has to take them into consideration while dodging direct attacks

3.5.4. TRASHBOTS



Health Points: 200
Damage: 40

Attack System: Leaks oil and then sets it on fire.

Main objective: The player must control its position in order to avoid the fire as seen on *Fig. 19*.

Fig. 19. *Trashbot trails set on fire*

3.5.5. WAVEBENDER

Health Points: 200
Damage: 40

Attack System: Creates pulses of energy that hurt the player

Main objective: The player must avoid corners as much as possible and never stop moving.

3.6. BOSS FIGHTS

Boss fights have become one of the most necessary events on an action game. Gigantic enemies throwing everything they have at the player. Kadick's Resolution has three different bosses. Each of them has a different type of attack pattern.

3.6.1. STAGE 1 BOSS

The first boss is based on the *Metal Gear Rex* from the Metal Gear Saga [6]. It is a gigantic robot that performs three different attack patterns. These attack patterns are displayed on Fig. 20.

1. **Aim and kill:** The boss aims at the player and shoots at its position. Dodging bullets becomes essential for survival. Fortunately, beating many Jolts on the way to the boss has served as a great training.

2. **Spread shooting:** The boss' turret shoot at the player using a spray pattern. This means that the turrets will be rotating from -45º to 45º while aiming at the player. This way it is better to keep calm and wait for the bullet to come instead of just trying to dodge bullets.

3. **Railgun.** The boss aims with lasers at the player as it loads the railgun. Once it has finished, it shoots two energy rays of energy at player's position. It is essential to keep moving. The rays are so radioactive that they keep on scene for some time until they vanish.

Fig. 20. *Stage 1 Boss Attack Patterns*

3.6.2. STAGE 2 BOSS

Second boss performs two different attack patterns at the same time, displayed on Fig. 21.

1. **Electronic Boomerangs.** Once the boss is presented, it deploys two boomerangs that will follow player's Y position on both right and left margins. The boomerangs will pass through the room damaging the player if they impact on him.

2. **Curve Laser.** There are two laser cannons displayed both on top and bottom margins. They follow the player's X position and every time the player passes between them they load and shoot. That curve will remain on its position, causing damage to the player as he passes through it. As the fight progresses, there will always be two different laser curves on display. Every time the cannon shoots, the first curve displayed will vanish and the new one will display. This process will repeat itself until the boss is beaten. This laser is made by generating a curve element which has several semi circumferences attached one to another forming the wave pattern. After that, there is generated a group of flares, the little balls of light that keep moving following the curve. These elements are followers, a special type of object in Phaser 3.

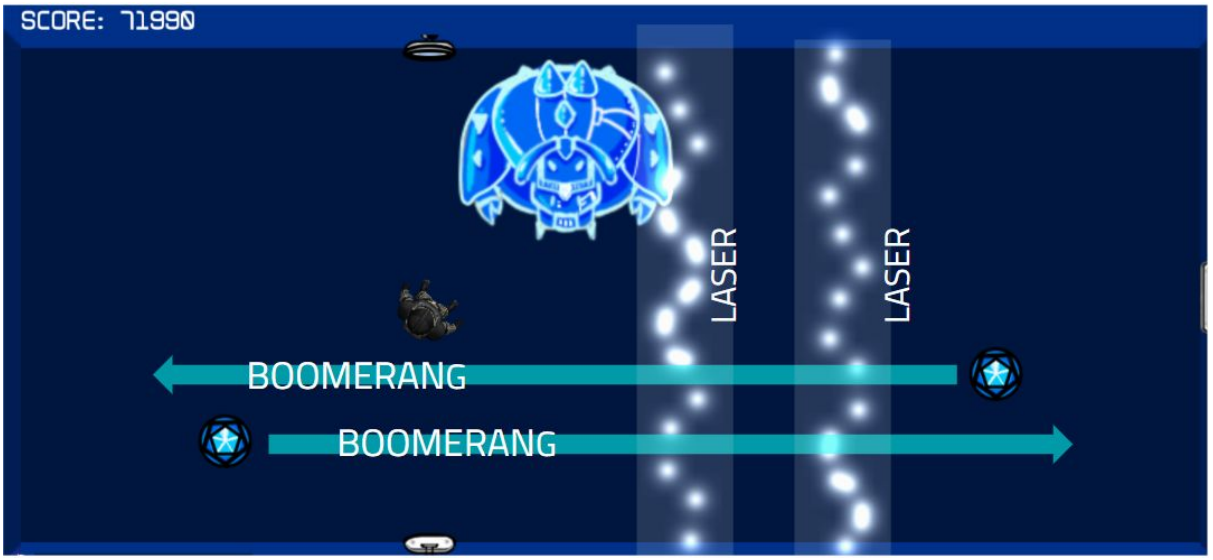


Fig. 21. *Stage 2 Boss attack patterns*

This boss has been developed using Phaser 3 tween animations that allowed designing fluid animations in outstanding performance without leaking memory.

3.6.3. STAGE 3 BOSS: ASIMOV

Asimov is the last boss of the game and is the most complex one. Its movement has been developed using Phaser 3 Timeline animations instead of just tweens. Each time Asimov reaches a keyframe, it deploys an specific attack pattern. Asimov is the only boss that perform two different phases displayed on Fig. 22.

1. **Phase 1**. On this phase, Asimov moves from one corner of the room to another while shooting. Its shots are limitless straight lines that form star-like patterns. There are two kind of attacks of this type:
    a. **Explosion.** This attack performs an eight direction array of lasers from Asimov's position. Those lasers move at an straight direction. To do so, there has been generated an array of sprites and they have been added a different angle and velocity.
    b. **Timed Spray**. This attack performs several explosion attacks at certain frequency with a rotation added. Thus, it performs a beautiful yet deadly floral-like pattern. Each sprite has been added an offset angle which increases for each timed iteration.

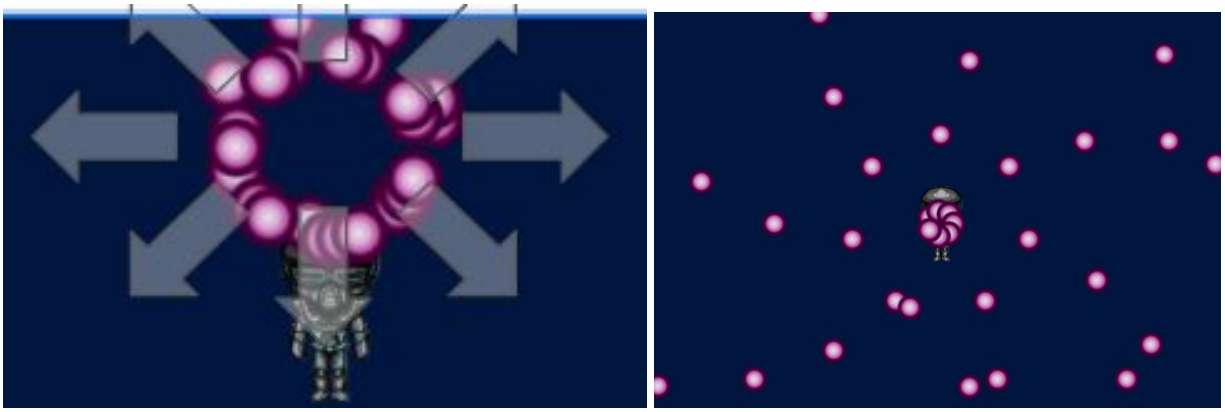


Fig. 22. Left: Explosion pattern. Right: Timed Spray pattern

2. Phase 2. On this phase, Asimov moves from one side to the other of the room performing the same two attacks from phase 2 but adding another pattern: Spiral explosion. It is similar to the basic explosion but bullets rotate around the explosion point while expanding generating an spiral. In order to do so there has been used a tween counter that performs a numeric increase in an specific time range. This counter is applied to a circle game object. Each time the game is updated, the circle increases its radius. In order for the bullets to rotate, first they have to be generated and placed on the circle. After that, they perform the RotateAroundDistance method that allows an item to orbit around certain object. Combining both elements there is a circle that gets larger while bullets rotate around its perimeter (Fig. 23). Thus, the spiral effect is made.

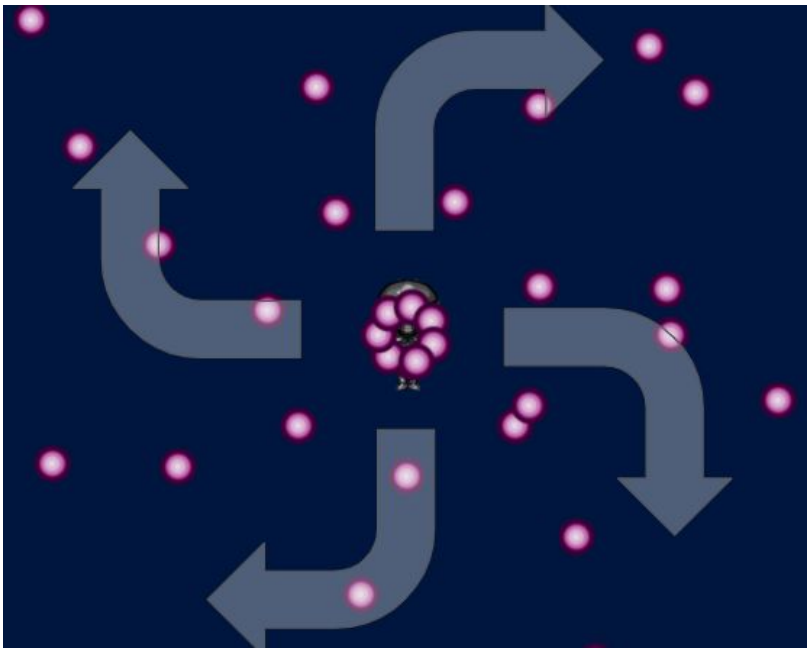


Fig. 23. *Asimov Attack Patterns Phase 2*

3.7. POWER UPS

There are four kinds of pickable items in Kadick's Resolution. Three of them are permanent power ups and one is a medikit. There are two random elements that affect the dropping of items.

1. Chance to drop item. There is a 30% chance for an enemy to drop any kind of item after being beaten.

2. Drop rates. Drop rates for each power up depend entirely on a second rate.
    a. Medikit: 70% rate.
    b. Attack Power Up: 10% rate.
    c. Fire Rate Power Up: 10% rate.
    d. Max Health Power Up: 10%% rate.

3.7.1. MEDIKIT

The most common pickable item. Heals 50% of Kadick's health. Health will not surpass its maximum value

3.7.2. ATTACK POWER UP

It boosts up attack indefinitely.
It is accumulative.

3.7.3. FIRE RATE POWER UP

It boosts up the fire rate of Kadick indefinitely.
It is accumulative.
The more power ups picked of this type, the more shoots per second

3.7.4. MAX HEALTH POWER UP

It boosts up maximum value for health and armor..
It is accumulative.

3.8. GAME UI

The UI was designed to be as simple as possible. While playing, there are four different panels of information:

1. **Score.** The score is presented as plain text on top (Fig. 24). The score goes up by hitting enemies or beating them.
    a. Hitting an enemy increases the score by 20 points.
    b. Beating an enemy increases the score by a certain amount of points.

Enemy	Points
Scancatcher	350
Jolt	600
Trashbot	900
Coulomb	1000

Enemy	Points
Wavebender	1200
Stage 1 Boss	5000
Stage 2 Boss	15000
Stage 3 Boss	50000

    c. If the player dies at any room and decides to start again, the score will be divided by 2.

Fig. 24. Score detail

2. **Armor bar.** This bar represents Kadick's armor capacity (Fig. 25). If the player gets hit, it will lose armor. If the armor reaches zero, the player will start losing health. The armor bar will refill itself when Kadick avoids getting hit for certain amount of time. The time for total recovery depends on the difficulty settings. The easier the game is, the faster the bar recovers.

Fig. 25. Armor bar detail

3. **Health bar.** This bar represents Kadick's health capacity (Fig. 26). If the player gets hit with no armor, the health bar will go down. If, at any given time, Kadick's life reaches zero, the game will be over and the continue screen will load. To regain health the player must collect medikits dropped by beaten enemies.

Fig. 26. Health bar detail

4. **Keys picked.** The keys picked by Kadick will be represented by a vertical array of icons (Fig. 27).

Fig. 27. Keys picked detail

The complete UI looks as displayed on Fig. 28.

Fig. 28. Complete UI detail

There is a minimap available for the player to use (Fig. 29). It is hidden until the player presses the TAB button. The red square represents the current room where the player is at. White squares represent completed rooms. Blue squares represent rooms that have not been visited yet. Green rooms represent rooms where there is a key. The golden square represents the boss chamber. Doors that connect rooms are displayed as orange squares.

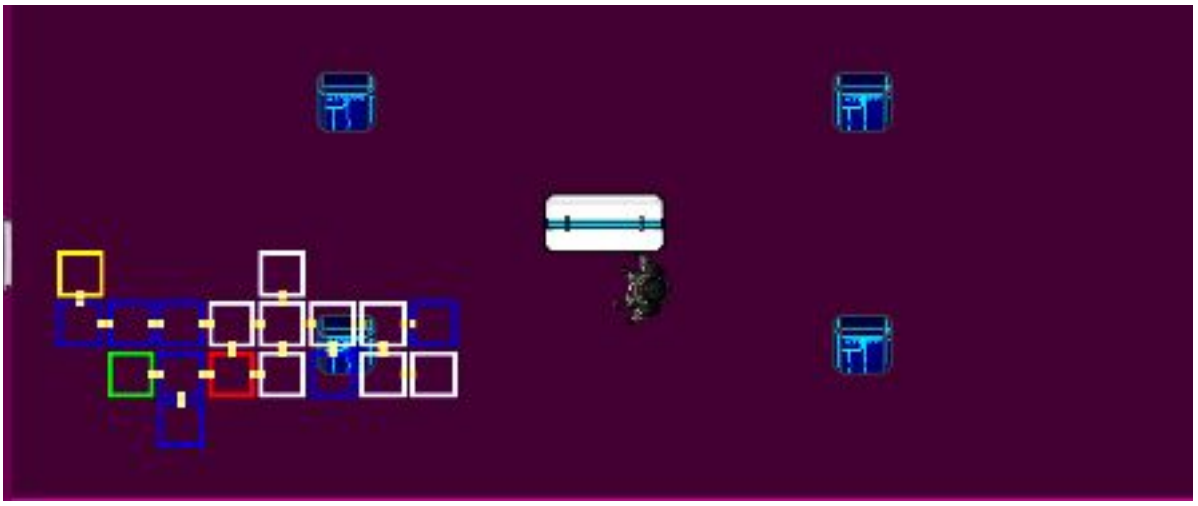


Fig. 29. *Minimap displayed over scene*

3.9. LOCALIZATION (I18N)

One of the main objectives of Kadick's Resolution is to reach as much audience as possible by localization. When this project began its development, it was decided to localize it in Spanish and English. Once the programming stage was finished and the textual content stage began it was decided to include Valencian as a language as a tribute to the Jaume I University.

To localize the game, a i18n folder is used, as seen on Fig. 30. There are three different JSON files that contain a hierarchy of key-value pairs.

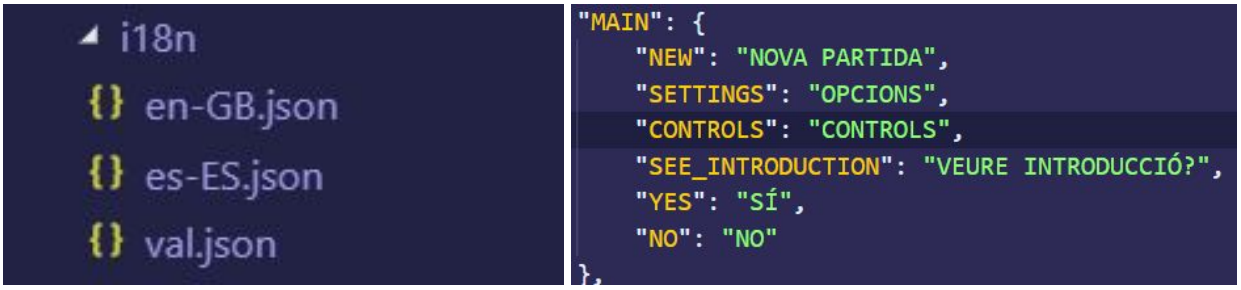


Fig. 30. *i18n files detail*

Selected language is shared during the game through the player data. If more languages were to be added, it would be just a matter of adding another JSON file to the i18n folder and allowing its selection on the language selection screen. Thus, localization is dynamically scalable for future releases. The player selects the language at the first screen (Fig. 31).

Fig. 31. *Language selection screen*

3.10. GAME START

The user starts at the language selection screen. Once the language has been confirmed, the main menu is loaded (Fig. 32).

Fig. 32. *Main menu screen*

There are three options: New Game, Settings and Controls.

By pressing the New Game button, the player is offered the choice to see the introduction of the game (Fig. 33). If the user chooses not to, the first screen will load immediately.

Fig. 33. *Introduction skip choice*

By pressing the Settings button, the player can select the difficulty settings for the entire run. The difficulty is set to NORMAL as default (Fig. 34).

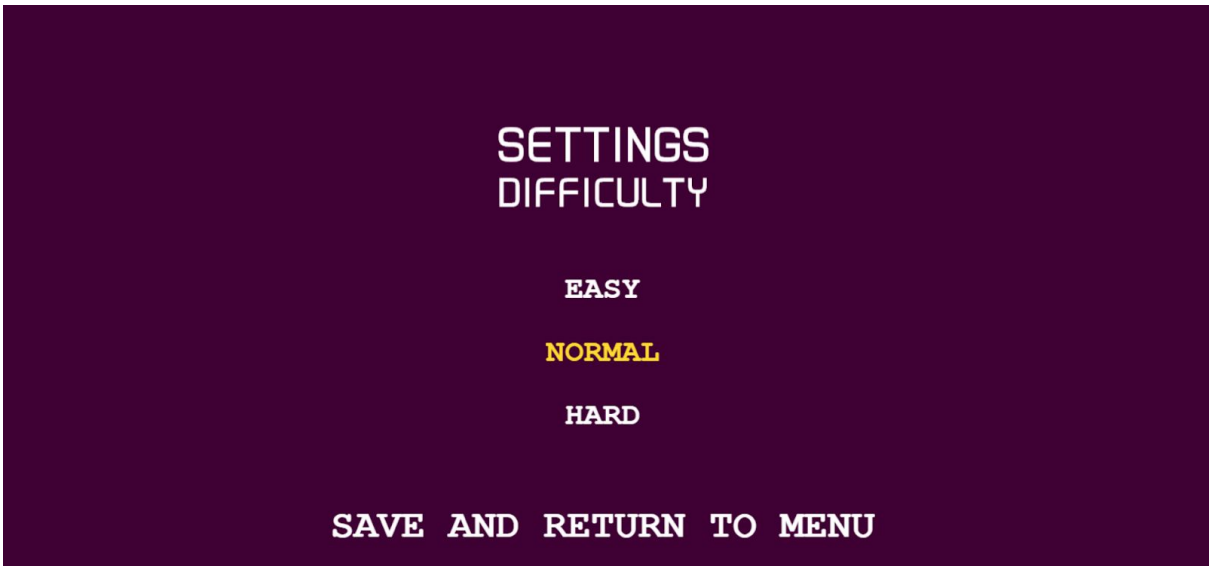


Fig. 34. *Settings screen*

The controls button shows how to control the player and basic information for the run. See 3.4 Player Controls for more information.

3.11. GAME FLOW

The game flow starts once the page is loaded.

First of all, a language has to be selected in order to the textual content to be translated. After that, the main manu will load giving us two options:

1. New game
2. Settings

Selecting the first one allows the player to start a new game. The player must select whether he wants to skip the introduction or not. Skipping the intro takes the player to the first level. Selecting not to starts the opening sequence. This is the point where the story is told through text on screen.

Whenever the player dies or once the game is completed, he is taken to the score screen, in which he is able to see his final score.

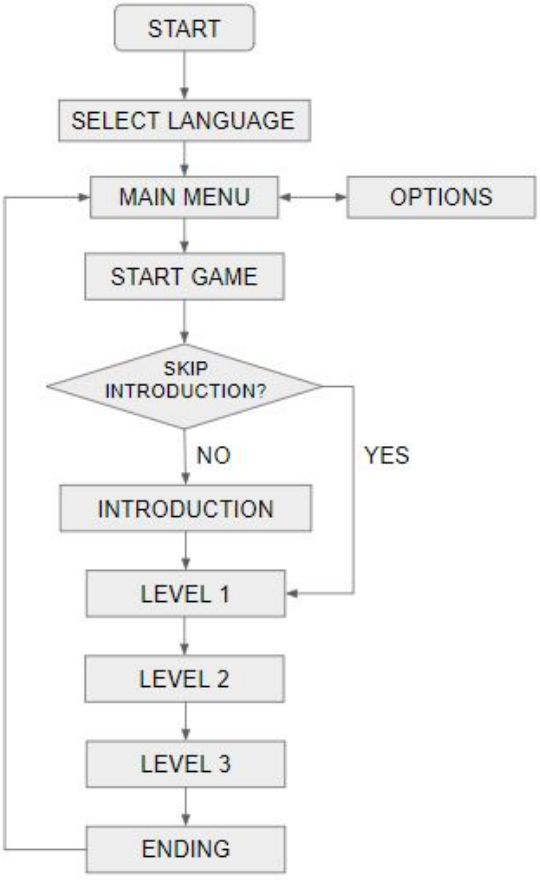


Fig. 35. *Game Flow Diagram*

The settings screen allows the player to change the difficulty at any moment.

Being killed at any moment takes the player to the continue screen (Fig. 36).

Fig. 36. *Continue Screen*

3.12. PROJECT DETAILS

The main characteristics of this project are displayed in the following table.

Lines of code	11244
Code Files	44
Sprites	83
Audio Files	19

4. RESULTS

It is expected that Kadick's Resolution game achieves the following results:

- Fully functional easy to use game.
- Responsive and attractive UI which allows the user to know at all time the player's status.
- Visual and extremely addictive gameplay with many enemies on screen
- Great Boss battles.
- Fully functional random scenario generated procedurally.
- Localized game with language selection.

In Kadick's Resolution the main goal was to use procedural design to generate pseudo-random content in order to improve the final experience for the user. The scenario generation is transparent to the user and its performance is outstanding as no memory leaks were found in the process. Thus, replayability is guaranteed as every run is different from the other. The result is at plain sight using the minimap UI, which allows the user not only to check at any moment the player's current location and main objectives but also to keep moving in the process. This capability was not included at first, but was found remarkably useful and was included in the development.

As this game is to be played on a web page, limitations in rendering and computing processes were taken into account. Fluid gameplay has been one of the main objectives since the beginning and the result is a player that moves fluently across the screen while shooting. The frames per second rate is stable throughout the full run.

Each enemy has a different pattern of attack and the player learns from them in order to train for the boss fights. Furthermore, there have been used different methods available from the Phaser 3 documentation to perform those patterns. Random spawns make the game unpredictable and as the run progresses, the feeling of gaining strength becomes tangible. The sense of awareness the first time an enemy is seen is right there. Kadick's Resolution is a game designed to sharpen the player's senses of position, movement and timing. Each boss fight is different from the other and requires extreme caution and attention to every single element on the scene to survive.

It is noticeable the urge to explore each stage to farm power ups and become more powerful. At this point the player faces two types of gameplay. The boss rush or fast completion, that involves running straight to green rooms in order to obtain keys as fast as possible and confront the boss fight with few power ups, which

makes them far more difficult. The other style suits better for the greedy player that loves the feeling of obliterating the boss by gaining so much power that it feels even unfair for them. Power up have been studied and designed to be likable to all kinds of audience. The drop rate is high enough to foster exploration but not too high to make the game boring. The variety of pickable items is enough to perform a noticeable difference between different kinds of gameplay.

The data structure is efficient and performs nicely throughout the run. Scene management performs fluently and the general gameplay is not affected by load screens.

Localisation was one of the main goals as it is essential to reach more audience. At the beginning of the development it was decided that localisation would implement Spanish and English language. It does so and is perfectly integrated inside the game data structure. Furthermore, the Valencian language has been added and the localisation folder is prepared to scalate very easily by adding more JSON language files. The effort to add a new language relays on adding another option to the language selection screen. After that, the language is spreaded across the game smoothly.

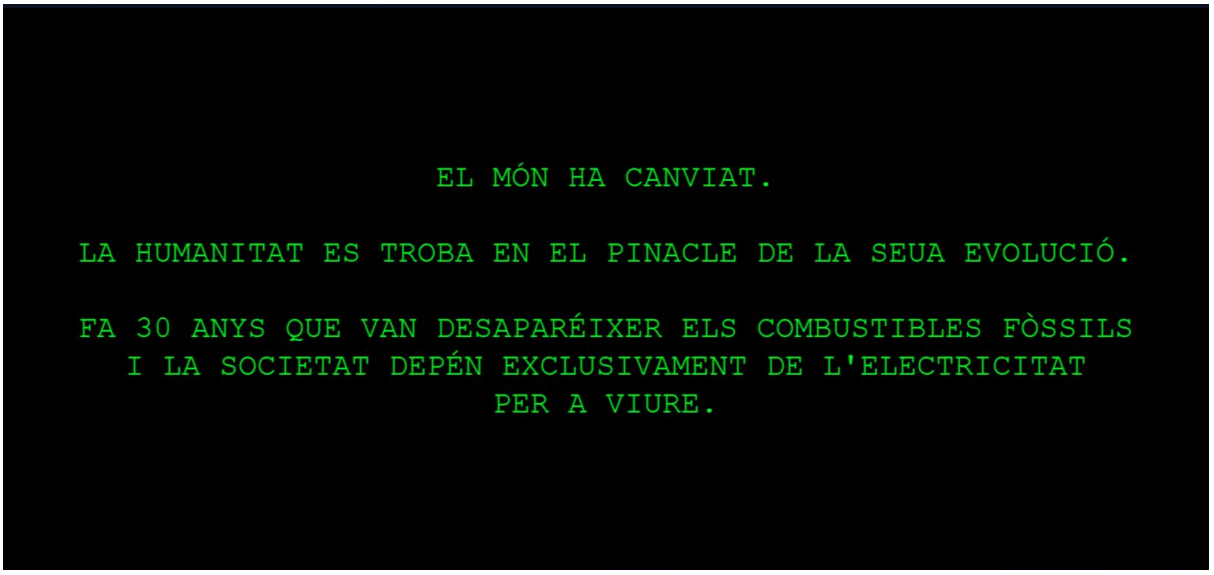


Fig. 37. *Localized textual content in Valencian*

The futuristic style has been taken into consideration at the design stage. Enemies share a futuristic resemblance to those which appear on books and movies. It was essential to create a cyberpunk-like atmosphere. By combining elements like modern art-deco elements found in modern laboratories on the props the user is transported to the next century. There have been used two types of typography: One that resembles a computer console and the other that shows the softness of more modern designs. Random distributions allow a rich display of different elements on scene.

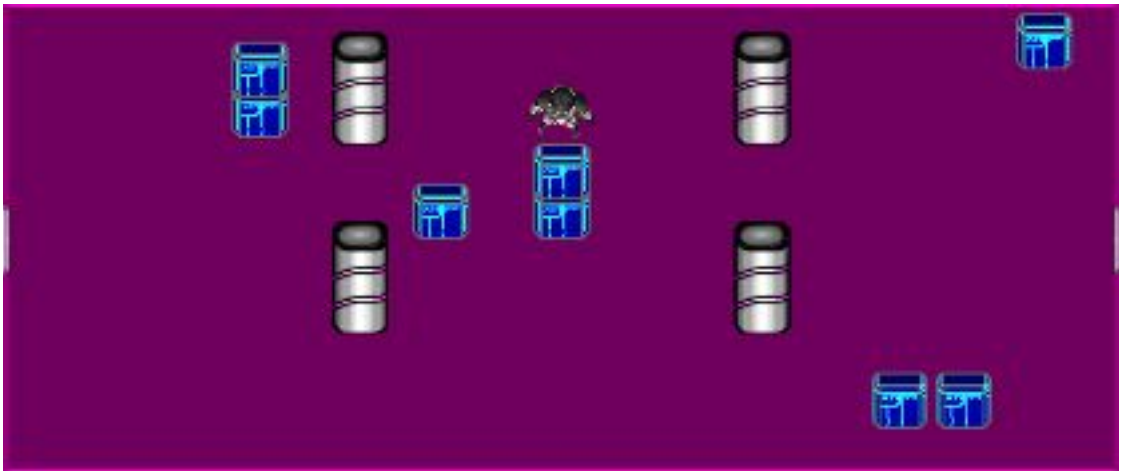


Fig. 38. *Scenario distribution example*

This page has been left intentionally in blank.

5. FUTURE PLANS

First of all, localisation would be one of the main goals for future releases. Including Italian, German and French would be essential to reach European audience and would be an easy task to complete.

The variety of enemies available at Kadick's Resolution is one of its weak points. There are five kind of minions and three bosses for three stages. It would be perfect to accomplish at least 5 types of different enemies per stage. In such manner, the player would not get bored of the same enemies over and over again and the sense of unpredictability would improve.

For future releases a special enemy would make its appearance: *The bounty robot*. This enemy would not hurt the player, but would try to flee from the room so there would be a time limit for the player to beat it. It would appear in all stages once and would always drop a power up and give a considerable amount of points for the high score to go up.

Another of the main goals for future releases is to add more boss fights and random mini-boss fights. This adds an extra difficult factor to the game. Mini-bosses will always drop a random power up and give a considerable amount of points for the high score to go up.

Asimov would still be the last boss but, it has been considered to set two fights against him. The first would take place after the third stage. It would be the same battle displayed on this project. However, Asimov would flee from the room after being beaten and the player would follow him through a long corridor while shooting and dodging bullets implementing a Touhou-like [7] gameplay (Fig. 39). After that, the player would have to get through more stages in order to reach to its hideout and beat him.

Laser turrets and traps have been designed but not implemented. Generating maze like random rooms is another goal for future releases. These rooms would be generated procedurally as well and would always drop a power up once completed.

Textual content relative to the plot is told by a sequence of text. It would be desirable to add animations so the player feels more identified with the protagonist.

An original soundtrack would be desirable since only audio effects have been implemented.

.

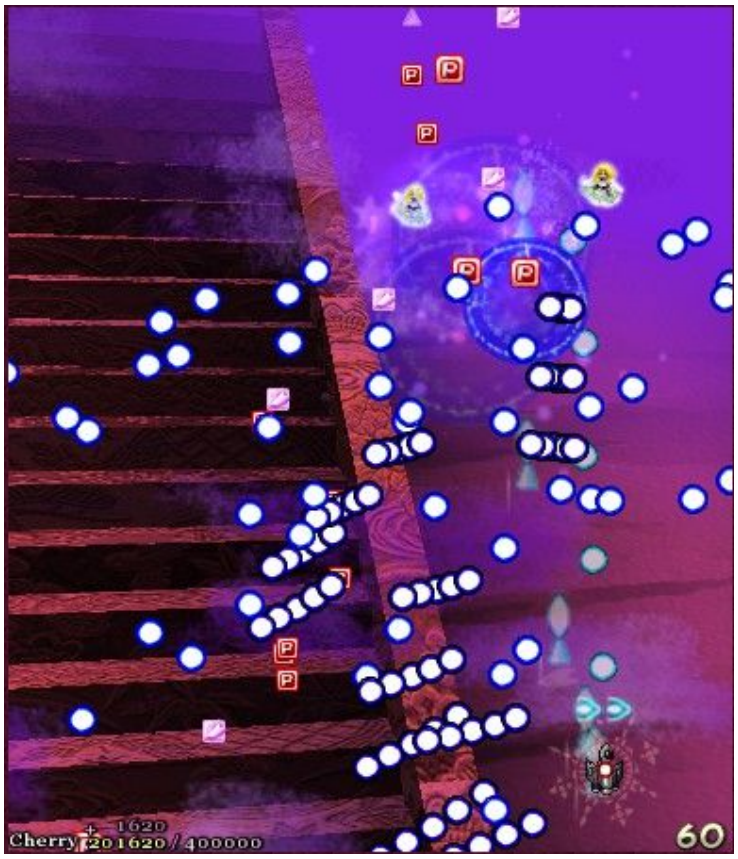


Fig. 39. Touhou gameplay

More mechanics would be implemented as well.

1. **Weapon change.** Kadick uses a blaster gun to defeat its enemies. However, adding more weapons would make the game more enjoyable and would add strategy to the mixture. To name some examples:
    a. **Shotgun:** A high impact short range attack that would pull the enemies back and would cause massive damage.
    b. **Charged shot:** Keeping the shoot button pressed would charge a laser bullet. The longer the charge, the more powerful the damage caused would be.
    c. **Railgun:** Massive shot with limited ammo. Would reload itself by picking this special ammo from defeated enemies.

2. **Bouncing bullets.** During the development of Kadick's Resolution the idea of using bouncing walls came to mind. Those walls would be easily identified by a color code and if they were hit, the bullet would bounce in angle. There would be a boss that covers itself and the only way to hit it would be attacking its back using the bouncing walls to do so.

3. **Exploding barrels.** These barrels would appear on screen and would explode on contact to any kind of laser, no matter friend or foe. Thus, enemies could be lured near them and then obliterated because of the explosion.

4. **Interactive floor panels.** The floor would have special kind of platforms that interact with the player in different forms:
    a. Electrified field. Would cause damage on contact.
    b. **Slippery platform.** Would cause the player to slide through it unable to control its movement.
    c. **Sticky platform.** Limits player's movement by certain factor.

This page has been left intentionally in blank.

6. TESTING

This section covers how this project has been tested. There is a brief representation of how useful unitary tests have been to test the scenario generation. User testing has been also absolutely decisive to improve the final experience.

6.1. GENERATION TESTING

The testing process began shortly after the start of development. In order to determine whether generation of scenario was successful or not, there were scripted some unitary tests. These unitary test work as a simulation that generates a new scenario 25 times in a matter of seconds. This makes it very easy to find weak points in the code and solve them. There were 4 main issues to take care of when generating the scenario:

1. There had to be exactly 3 keys.
2. Start point could not be a key room.
3. The boss chamber had to be generated and accessible.
4. Keys had to be located in random rooms and none of them could be the start room nor the boss chamber

These issues were evaluated individually every iteration (Fig. 40).

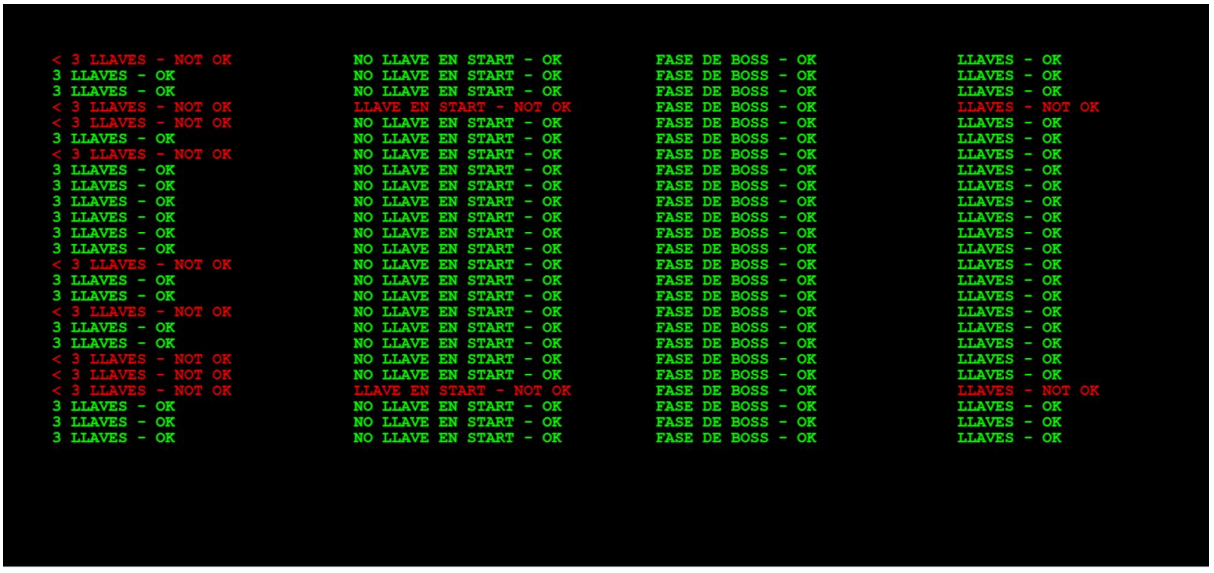


Fig. 40. *Unitary tests: Finding weak points*

6.2. GAMEPLAY TESTING

The gameplay testing consists on an iterative process that started when the attack system for both enemies and player were developed. As the physics system implemented in this game is Arcade, the bounding boxes are rectangular shaped. In this context, there is a lot of trial and error process in order to adjust the bounding box sizes and making them flow nicely with enemy movement and scenario distribution. Taking into consideration that the gameplay is mostly based in shooting, collisions between lasers coming from players and enemies had to be tested. Lasers had to disappear when hitting walls, characters and props (Fig. 41).

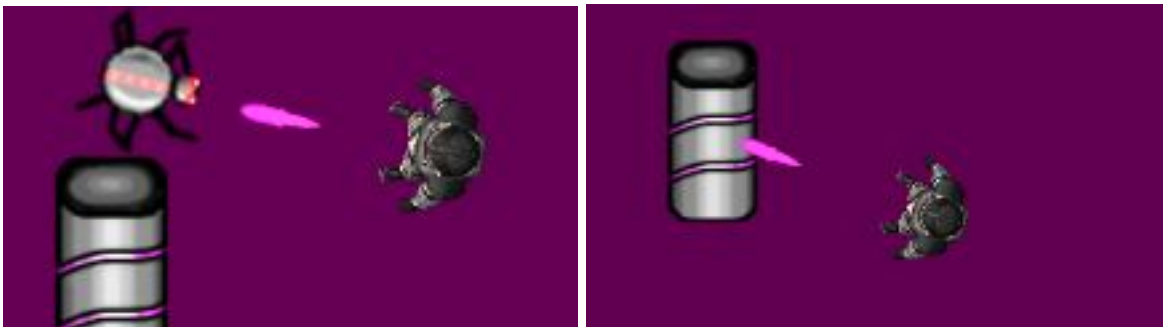


Fig. 41. *Collision testing: Enemies and Scenario Props*

Scenario distribution had to be tested too, as the player had to move fluently among props no matter what the resolution would be. In order to do so, the game has been tried in rectangular screens and square screens (Fig. 42).

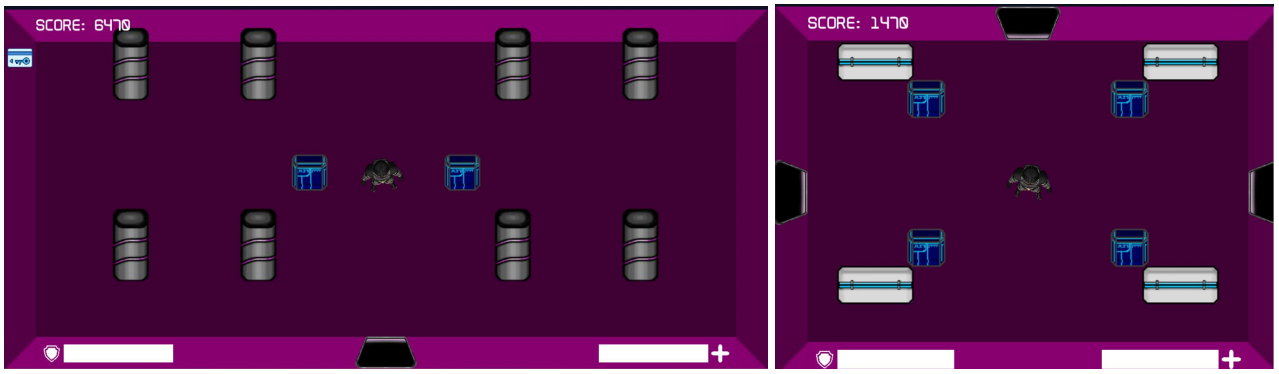


Fig. 42. *Scenario distribution testing*

Minor errors were solved during the process, specially the collision system tended to fail with some types of enemy and was later improved. The supervisor of this project also gave extremely helpful feedback as he tried the game from other perspectives. The user experience was enhanced and some characteristics like the dynamic resolution adaptation were introduced.

6.3. USER TESTING

After having solved these issues, it was clear that some other errors would be avoided as the developer knows his code and where not to look. Thus, it was essential to have other people try Kadick's Resolution and learn from their feedback.

Some friends had been asked to try the game along the development process. They were given finished tasks milestones and were told what to try but not how. As the game kept advancing, many minor errors and bugs were found and solved.

Once the project was getting to it final stage, other group of people was told to try the game. It was essential that they did not know anything about it so their perspective was unaffected by experience. They were asked if they found the UI too simple or too invasive and changes as the disposition of the armor and health bar were made. They thought that its icons were barely visible (Fig. 43) so the bars were disposed separately and at the final stage, as explained before, the armor bar remains at the bottom left and the health bar at the bottom right of the screen.

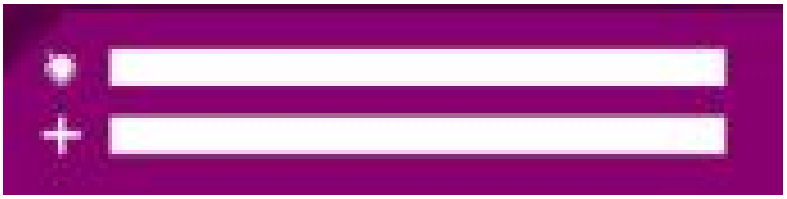


Fig. 43. *Initial Armor and health bars*

Difficulty settings were tested with people that had little experience in video games of this matter. Normal setting should be challenging but not too much as it is the default option. Easy mode should not be too easy and Hard mode should make it possible to complete the game.

Lastly, they were asked what they thought about the game in general perspective. Their opinion overall is summarized in these points.

1. The game is not boring at all. It is interesting and pushes the player to explore the stages.
2. Enemy fights are fair and possible to beat.
3. Boss fights are really interesting taking into consideration that they perform attack patterns from different genres.
4. They were not interested in the plot at all.

This page has been left intentionally in blank.

7. CONCLUSIONS

The procedural generation of scenario using a JavaScript framework has been a challenge that involved practical algorithms learned during the degree. It really is inspiring to see the final result as it is available to check at any moment. Stages are generated both horizontally and vertically and the distribution system adds depth to the style of the game by offering cover.

The gameplay is as easy to understand as fluid. It allows dodging and attacking perfectly. The performance of the browser is not affected even with dozens of bullets at the same time.

Many types of gameplay coexist on the same game. Running straight to fight bosses or gaining as much power as possible in order to humiliate the beasts are an available choice for the player to make. The score system adds an extra challenge as the player can try to surpass the previous high score and has to avoid dying to do so. Death penalization adds an extra risk to the gameplay as the user has to traverse through the dungeon all over again and loses half of the score points.

The enemy development was other challenge to take care of. Attack design come from many references from arcade games and have been introduced smoothly. From the easy ones like the Scancatcher that follows the player around to the complex attack patterns of Asimov, each one of them has been carefully designed and implemented in other to push Phaser 3 to its limits.

The power up system works like a charm as a game flow manager. It sets both the sense of growth and the sense of challenge. Trying to pick every single power up available at any stage reinforces the will for exploration.

The audio has been meticulously selected and edited to fit perfectly into the plot and the main gameplay, generating an immersive atmosphere.

Kadick's Resolution has become an instructive and challenging adventure. It has gone beyond its initial expectations offering an immersive experience and outstanding gameplay. Exploiting the limits of the Phaser 3 API has been really interesting as new forms of confronting certain situations have been found and tested.

This page has been left intentionally in blank.

8. REFERENCES

Kadick's Resolution takes its references from authors of the world of science-fiction and cyberpunk. The main character: Kadick, takes its name from Philip K. Dick, author of "Do Androids Dream of Electric Sheep?", "The Man in the High Castle" and "Minority Report" among others.

The antagonist, Asimov, takes his name from Isaac Asimov, author of a vast number of science-fiction futuristic books like "I, Robot" and the "Foundation" serie. He is also the author of the three laws of robotics.

The gameplay is based in games such as "The Legend of Zelda: Link's Awakening" (Nintendo, 1993, Fig. 44), "The Legend of Zelda: A link to the past" (Nintendo, 1991, Fig. 47), "The Binding of Isaac" (Edmund McMillen, 2011, Fig. 45) and "Metal Gear" (Konami, 1987, Fig. 46).

Fig. 44. *Legend of Zelda: Link's Awakening*

Fig 45. *The Binding of Isaac*

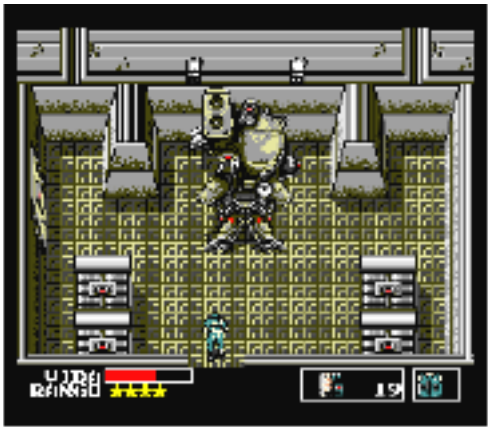


Fig 46. *Metal Gear*

Fig 47. *Legend of Zelda: A link to the past*

The art references come from games like "Cyberpunk 2077" (CD Projekt, 2019, Fig. 49) and "Nier Automata" (Platinum Games, 2017, Fig. 50). There are also references from movies like "Blade Runner" (Fig. 51), based in "*Do Androids Dream of Electric Sheep?" (Philip K. Dick) and "Battle Angel Alita" (Manga version by Yukito Kishiro, 1991 - 1995, Fig. 48).*

Fig. 48. *Battle Angel Alita*

Fig. 49. *Cyberpunk 2077*

Fig. 50. *Nier Automata*

Fig. 51. *Blade Runner*

9. BIBLIOGRAPHY

Programming

[1] Phaser API Documentation website: https://photonstorm.github.io/phaser3-docs/
[2] Phaser 3 Examples website: https://labs.phaser.io/
[3] Stack overflow website: https://stackoverflow.com/

Software

[1] Visual Studio code website: https://visualstudio.microsoft.com/
[2] Photoshop website: https://www.adobe.com/es/products/photoshop.html

Videogames

[1] Diablo 3 website: https://eu.diablo3.com/es/
[2] Official Blizzard website: https://www.blizzard.com/es-es/
[3] Official Minecraft website: https://www.minecraft.net/es-es/
[4] Official Terraria website: https://terraria.org/
[5] Official The binding of isaac website: https://bindingofisaac.com/
[6] Official Metal Gear Solid website: https://www.konami.com/mg/mgs5/tpp/en/
[7] Touhou website: https://moriyashrine.org/

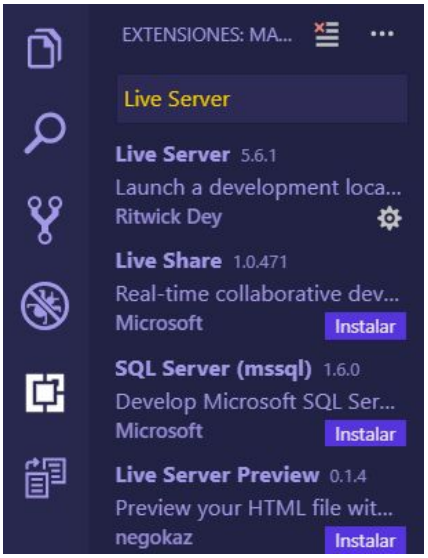
This page has been left intentionally in blank.

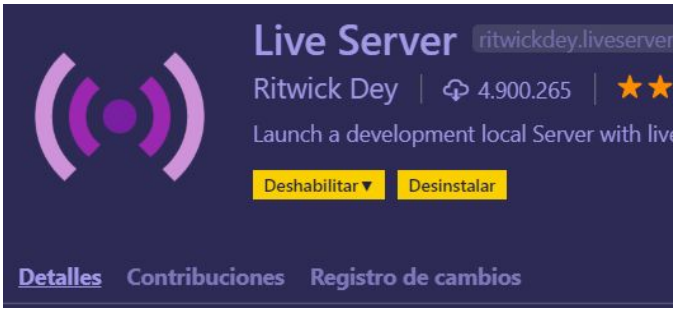
10. USER'S MANUAL

1. To download this project, the user has to clone the project from the following github URL:

   Github url: https://github.com/ManuSenpai/KadicksResolution

2. Once cloned into the user's computer, it must be opened using Visual Studio.
3. After being opened, the user has to install the "Live Server" extension.





4. Once the extension has been installed, the user must press the "Go Live" button at the bottom right.

5. After that the game will load automatically.

6. If it does not load, the user must enter this direction into the internet browser: http://127.0.0.1:5500/

7. Gameplay instructions:
    - **Move right: Key D.**
    - **Move left: Key A.**
    - **Move up: Key W.**
    - **Move down: Key S.**
    - **Minimap: Tab Button.**
    - **Shoot: Left click.**

8. This game is also available to play directly at this URL: http://teseo.act.uji.es/~al315320/Kadicks%20Resolution/

9. This game works best on Google Chrome.