Silent Specter Documentation

This document contains information regarding the completion of an individual project to create a classic Game Boy game. This project was conducted by Valentin PATELLA, a second-year student in the BTS Game Programming & Game Design program at the Lycée des Arts et Métiers, Luxembourg.

Table of contents

Game overview

Concept

Story

Silent Specter is a 2D Gameboy Classic game in which you play a little ghost who has forgotten the reasons for his death. To find answers, he will explore the places he frequented in life, collecting artifacts and objects from his past. The mystery will unfold as he gathers these objects.

However, being a ghost is far from easy and risk-free: in its explorations, the ghost must avoid losing its spectral integrity, which is diminished by attacks from other ghosts.

But ghost life does offer a useful little power: you can pass through certain walls and avoid attacks from some ghosts using your invisibility!

Genre

Silent Specter is a platformer/puzzle game.

Target audience

The target audience for my game would be 12+. The graphics are neither bloody nor shocking, but some story parts contain mature topics. However, any player can enjoy my game, as it is neither easy nor hard.

Gameplay

Core mechanics

In the game, players can perform different actions:

- Jump
- Use invisibility ability
- Move

Controls

• D-pad: move to the left/right (left/right arrows), fall from a platform (down arrow), interact with level 2 stairs (up arrow).

• A button: activate invisibility ability.

• B button: jump.

• START button: Pause Menu.

• SELECT button: open Collectibles Menu.

Win/Lose conditions

The only "win" condition in the game is to complete all the levels. Even if the player misses some collectibles, they can complete the game even if they will probably miss some points in the story.

The "lose" condition is primarily the "disappearance" of the ghost, losing all its spectral integrity, resulting in a Game Over.

Game loop

The main game loop in Silent Specter is exploring, jump and parkour around diverse levels, collect objects to understand parts of the story and avoid getting hit by ghost enemies.

Player replayability

If a player completes the game 100%, meaning they find all the collectibles, the game, in its essence as an investigation and puzzle game, should not be replayable.

The only reason a player would replay my game would be if they tried to find all the collectibles they missed during previous sessions.

Level & world design

Numbers of levels

In Silent Specter, the player will have to explore four levels, representing four distinct places the ghost used to live in when he was alive. Those levels are:

• The House: the ghost house in which he lived.

• The Shop: the store where the ghost used to shop when he was alive. This level introduces a new mechanic.

• The Office: The ghost's workplace. Something happened in this place...

• The Hospital: the ghost regularly visited this place, but why?

Level progression

To progress in each level, players will have to explore and find the exit of each level (located towards the right edge of the level). Triggers are used for the exit, leading to the next level.

Development

Mechanics/Features

In this section, I will explain how I have developed the mechanics of the game, why some got added/modified/removed from the original plan, how I have implemented them, etc.

This will not concern the mechanics like basic movements (jumping, move...).

Player related

Invisibility

The ability to become invisible was the only ability I had in mind for the ghost. It made the most sense in the context of my game and the story. Developing it was not the most difficult part.

It consists of a boolean value called 'bIsInvisible' that changes when the player presses the A button. This value is notably used to pass through cracked walls or one of the types of enemies.

This ability was the first one I implemented, and I did not have much difficulty implementing it. I just had to modify the behavior of the basic A button by attaching a script to that button.

Health

For the player's life, the basic idea was to have 'spectral integrity', one of the only ways for me to justify the fact that the ghost could die again.

This system consists of a 'Player Health' value which is initialized to 4. Each time an enemy hit the player, their health will be reduced by 1. Once this value reaches 0, the ghost 'vanishes' and the death screen is displayed.

For the display of health, I used an actor called HUD_Ghost pinned to the screen (top-left corner), whose sprite changes according to the value of Player Health.

This feature was quite simple to program, however I had to think about how to manage the Player Health value during scene changes (especially in the second level), at the beginning of each level or when the player restarts the game after a death.

The HUD_Ghost is updated using an Update HUD script, which is called whenever the player's health needs to be modified. This script consists of a switch that executes specific code based on the Player Health value at the time the script is called.

Heal Orb

When implementing the first enemies, I had to think about adding collectibles that would allow the player to regain integrity. That is why I designed small energy orbs which, once collected, disappear and restore health (1) to the ghost.

The logic is quite simple, but I had to think about how to responsibly manage the update of the HUD_Ghost each time a Heal Orb is picked up.

Enemies

Initially, my idea for the development was to create a type of enemy ghost inspired by the Boos from Mario. Ghosts that would have two behaviors:

- Chase the player when their back is turned.
- Move away from the player when they are looking at the ghost (unlike the Boos that just stop moving).

I ran several tests to implement this type of enemy, but I could not make it feel natural. I managed to get the enemy, once on the player's screen, to check if the player was looking in its direction. However, the ghost would move in the opposite direction from the player, but once the player changed direction, the ghost would continue moving away. Also, when the player got close to the enemy, it would get stuck between two states, causing significant lag.

Ultimately, I decided to remove this type of enemy from the game, but I will keep the idea in mind for future projects, if the opportunity arises! But in the end, I decided to pay tribute to this enemy who left too soon by keeping the name Scared Ghost for one of my enemy types present in the definitive version of the game!

Scared Ghost

This type of enemy relies primarily on the player's ability to become invisible. Indeed, to avoid being hit by this enemy, the player must be invisible to pass through it.

The player can, however, avoid this enemy by jumping over it (in the spirit of a classic platformer where there is not just one way to dodge enemies).

In certain sections of levels, it is impossible to avoid this enemy without using invisibility, forcing the player to carefully time the activation of their power. If a Scared Ghost ever hits the player, their integrity is reduced by 1. A small cooldown is present to prevent

repeated damage. This enemy's movement is based solely on patrol points towards which the Scared Ghost moves horizontally. It does not chase the player upon sight.

I decided to create a 'ScaredGhostPatrolling' script which requires as parameters the X coordinates of the two patrol points and the Y coordinate (the same for both points as I am only making a horizontal movement). The intention behind this script was to group the patrol logic that is common to each enemy of this type. I only had to specify the coordinates of the patrol points for each enemy I placed in my levels. It was also a way to reduce the lag and the use of ROM associated with the OnUpdate event.

Invisible Ghost

This second type of enemy relies on its ability to transition between invisible and visible states at intervals. It made sense to give this invisibility ability to another type of enemy as well.

The player can only pass through this enemy without taking damage when it is in its 'invisible' state.

The programming logic behind this enemy is fairly simple but requires some adjustments. For the 'invisible/visible' phase system, I use the OnUpdate event, with Wait statements representing the transition between the enemy's two phases. Depending on the phase, I change the enemy's sprite. At the transition between the two phases, I apply a 0.5-second Flicker effect.

Finally, in the OnHit event, I follow the same logic as with my first enemy type, but I check if the enemy is in the invisible phase (and therefore cannot deal damage) or in the visible phase (and therefore deals damage). The one small thing I had to pay attention to was when starting the level: the enemy had its sprite representing its invisibility phase but still caused damage, but I quickly fixed this by setting bIsGhostInvisible to false OnInit.

I am satisfied with the two types of enemies present in the definitive version of the game, even though I had other ideas for enemies besides ghosts, such as:

• Salt placed on the ground (which, according to widespread belief, wards off ghosts) that would also cause damage to the player. This raised some concerns for me, particularly regarding the art and lore of the game. As far as art goes, drawing grains of salt seems easy, but it really did not look good on the Gameboy screen during my tests. As for the narrative, it did not make much sense; the living people were not fighting ghosts, and there was not any exorcism story or anything like that.

• Incense diffusers: this idea would have allowed me to have an element of the scenery that would not inflict damage per se on the player but rather a debuff on their invisibility or movement ability (inability to be invisible or reduction of jump height/slowing of movement).

Representing this was not a problem, however, the story itself was another issue, as there was no exorcism or ghost-hunting element in the various levels.

What would have been most challenging for me was the zone system in which the incense would have been effective.

One of the mechanics I tried most was reducing the ghost's movement speed by modifying the WalkVelocity value in GBStudio (using the Engine Field Update event), but ultimately, once it was reduced, I couldn't reset it to default, or it would decrease in a loop until there was no movement speed at all when the player was within the zone of effect.

Environmental elements

Cracked Walls (Invisible Wall)

This element was one of the first ideas I had when thinking about the invisibility system, allowing the player to pass through walls (as a ghost is supposed to be able to do in widespread belief). To 'justify' the fact that some walls were passable, I decided to add small cracks and holes to their sprites to clearly distinguish the 'solid' walls from the 'passable' ones.

The wall's logic simply involves detecting the player's collision with the wall (in the OnHit event) and checking the player's ‘bIsInvisible’ variable at the time of the collision. If the player is invisible, they can pass through the wall (deactivate the collision of the wall); otherwise, the collision box prevents them.

In the third level, I tried a variation on these walls by rotating them horizontally, but unfortunately, their collision detection with the player was not working properly, preventing the player from passing through them correctly. During testing, I could pass through them from below but not from above. However, to pass through from below, the player needed very precise timing for the collision to be detected between the two actors. I tried increasing the wall's collision box, but that did not work very well either, so I abandoned the idea.

Stairs (Level 2)

In the second level of the game, I introduced the staircase mechanic. The logic behind it is quite simple: by using the up arrow at one of the staircases, the player is 'teleported' to another staircase.

The idea behind this mechanic was to create a sense of 'labyrinth' in the level, so that the player would not really know where they would end up when taking a staircase.

To achieve this result, I override the behavior of the up-arrow key. When the player is inside a trigger representing the interactive area of the stairs, the variable 'bUseStairs' is set to true. Pressing the up arrow key teleports the player to the exit staircase (its coordinates are stored in two global variables, ExitStairsX and ExitStairsY) linked to the entrance staircase. During this interaction, a script called 'UseStairs' uses GBStudio's Change Scene event, sending the player to the coordinates of the exit staircase.

One of the issues I had with this system was that the player's health was reset to 4 every time they used a staircase. This happened because, initially, I was resetting the Player Health variable to 4 every time a level was launched. This was not a problem until I implemented the staircase system. To compensate for this, I decided to reset the Player Health value to 4 only when changing levels (i.e., in the level exit triggers). It was a minor issue, and I fixed it easily.

Locked Door

I decided to add a locked door system for certain levels. They cannot be passed through while invisible, forcing the player to find the key within the level, thus encouraging exploration. Perhaps they don't really make sense in the context of the game, but it's one of the only ways I have found to force exploration. Once the player retrieves the key found in the level, the door unlocks (OnHit event) and allows the player to continue. The logic is roughly the same as invisible walls, but it just checks if the player has the key and not if they are invisible.

Collectibles

Collectibles were also one of the first ideas I had for the game. Small objects that, once collected, reveal information about the story.

Each time an object is collected, a short dialogue appears on the screen. Once collected, an object is visible in the 'Collectibles' screen. Collecting objects is not mandatory (except for keys), but it really helps the player understand the ghost's story.

If a player does not collect all the objects, it is not a problem; they can still finish the game. There is no ending based on the number of objects collected. Once an item is retrieved, it simply disappears from the level. The idea of having multiple endings

depending on the number of collectibles was in my mind from the start, but I have not found any suitable ending ideas, but it could be a possibility if I rework the game!

Graphisms and arts

To avoid repeating the same mistakes I made in a previous project, I decided to use free assets found online. So, I did some research and found assets for GBStudio from creators on Itch.io. I tried placing them in my levels, until I realized they were not suitable at all for my game's 2D sideview; many of them were designed for top-down games.

Ultimately, after noticing this, I decided to create the assets and sprites for the different characters myself. The effort was clearly reduced, given the limited detail and the small size of the assets for a Game Boy game. It was particularly good exercise for, let us say, improving my artistic skills.

However, I still had to decide to use the power of artificial intelligence to generate certain assets, particularly the various backgrounds for screens like the Home Screen, Main Menu, and Death Screen.

To create these screens, I asked ChatGPT or Gemini to generate an image in the Game Boy style, resized these images on a website (https://imageresizer.com/bulk-resize) to the Game Boy screen size (160x144px), and then used the PixelLab plugin in Aseprite to modify some details such as the text, the colors used (to match the Game Boy palette), etc.

To reduce lag and ROM usage, I had to manually reduce the amount of detail in the background, thus reducing the number of unique tiles. The Collectibles and Controls screens were made by me (it was pretty simple, obviously). For the sprites and assets, everything was created by me, using Aseprite. I specifically used tilesets to arrange the different elements and generate the map in Tiled.

At the beginning of the project, I researched which color palette to use and the required asset sizes. Funny enough, at a certain point in development, I forgot that one of the colors absolutely should not be used for the actor sprites, forcing me to change it for each sprite, but the change was made very quickly.

Additional remarks

During the game's development, I experienced many moments of doubt and struggle. One of the main problems was managing ROM memory, unique tiles, and map creation with Tiled and Aseprite (primarily concerning map size).

Regarding memory management, during development, the debugger in GBStudio indicated high ROM usage. From the very beginning of the project and the implementation of the first board and its mechanisms, I had already used almost 100 KiB of ROM. This worried me greatly; I was experiencing significant lag and performance drops. I think one of the main reasons was using the Platformer+ plugin for GBStudio (to be confirmed), and that I was using far too many unique tiles at the beginning of development. As development progressed, realizing that too much detail could be the cause of these performance losses, I greatly reduced the number of unique tiles!

Finally, at some point, I realized that the OnUpdate events were the main source of lag, so I made sure to reduce them as much as possible.

I tried my best to minimize ROM usage and I am quite satisfied with the game's performance. However, in some parts of the levels, the game lags a bit, but strangely, it is less noticeable on Gameboy than in GBStudio (or in the web version on Itch.io).

To be honest, there was a long period when I could not work on the project and development stopped for a few weeks, but in the end, once I got back into development, I was excited to continue and finish the game!

Final product

The final version of the game is available on my itch.io profile! The game fits perfectly for me, in my vision of a puzzle/mystery game with a story that unfolds as you explore, through collected objects or even level design and scenery. The game's difficulty is not too high, making it enjoyable to play and, I think, fits well into this movement of creating Gameboy games with a modern look, both in terms of art and mechanics!

I think the game could benefit from additions, perhaps a sequel, with more mechanics and story! In the end, despite a development process that was sometimes a little complicated, I am very proud of the product I delivered, and I look forward to the various comments and reactions of the people who will play it!

Credits

Assets

Mostly created by me, using Aseprite. The main menus and transition screen were generated by AI and slightly edited by the PixelLab plugin or by me.

SFX

The sounds that can be heard in the game are all from a sound pack created and made available by Tronimal on itch.io (https://yogi-tronimal.itch.io/gbfx). Thank you to them, the sounds fit perfectly into the game's world, and the number of sounds available in the pack is impressive!

Music

The two music tracks in the game (one throughout and one during the ending screen) come from an 8-bit music pack created by Tiptoptomcat on itch.io (https://tiptoptomcat.itch.io/8-bit-gameboy-songs-gb-studio). This pack was one of the few (besides other packs by the same artist) that contained sounds in the .mod format (necessary for use in GBStudio). Thanks to them, the music perfectly matches the atmosphere I wanted in my game!

I tried converting other music from different packs (.wav to .mod) but it is nearly impossible, as the .mod format is extremely specific.

List of software used

Art and map:

- Aseprite (with PixelLab integrated plugin)
- Tiled
- ChatGPT or Gemini (for AI-generated screens)

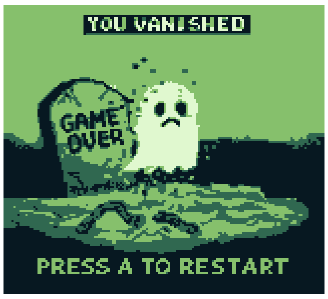
Programmation

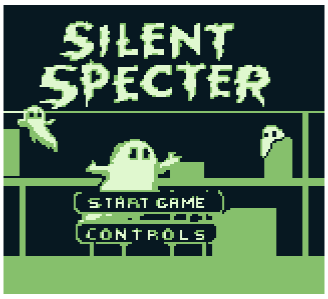
- GBStudio (with PlatformerPlus plugin for collision management, additional platformers functionalities)

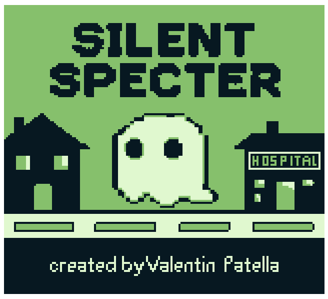
Some visuals

In-game sprites/art

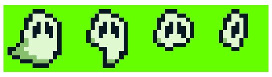
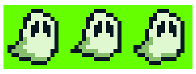
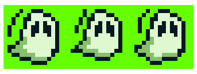
Screens:



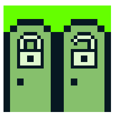




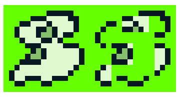
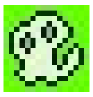
Player-related:

Environmental elements:



Enemies:

Cemetery of idea: