Report NDVW - When We All Fall Asleep

Giacomo Sanguin, Giacomo Schiavo

1. Introduction

"When We All Fall Asleep" is a single-player horror game in first person view. The player wakes up in a dark cave, trapped and desperate to escape. The cave is populated by a monster, named Akumu, whose only goal is to kill everyone he encounters in his path. To find a way out, the player must collect five different items scattered throughout the cave. These items hold the key to unlocking the exit and surviving the terrifying ordeal.

1.1. The Story

"When We All Fall Asleep" is a horror game set in a dark cave, where the player, waking up with amnesia, must collect items to survive and escape. As the player retrieves the items, an intricate plot unfolds. The protagonist, a mafia member named Tom, murdered a debt-ridden family, later falling into a coma after a car accident. In the cave, the player relives memories, discovering he is the killer of the family. Each room represents a victim, and the murderer is Tom.

The story of unfolds through the letters and diaries scattered throughout the cave, each revealing a chapter of the tragic narrative:

1. **Father's Room**: the opening letter reveals the father's debt to the mafia, leading the family to separate for protection. The father, committed to resolving the situation, expresses remorse for his actions, mentioning a "monster" that seems to be Tom, the killer.
2. **Daughter's Room**: the diary of the little daughter reveals the absence of the father and happy memories when both parents used to read her stories before bedtime. It also hints at a "monster" that now appears to be present in the cave.
3. **Wife's Room**: the wife, feeling spied upon, wishes to return to her husband to feel safer. This underscores their lack of security and the desire to reunite.

4. **Boss's Room**: the boss's diary reveals that the family, monitored by Tom, was destined to die after the father repaid the debt. Tom becomes the killer tasked with this terrible mission.
5. **Tom's Room**: Tom's letter reveals that after killing the family, he was hit by a car and fell into a coma. The cave represents his purgatory, where he must decide whether he deserves life after the atrocities committed. The final letter, written by Tom's subconscious, reveals that the protagonist is Tom himself, now in a coma and grappling with his own subconscious. The story concludes with a reflection on justice and the value of life, leaving the player to decide Tom's fate.
   The letters and diaries are not placed in their respective rooms. This decision was made to maintain narrative consistency and build a climax towards the end with the last letter. Additionally, the labyrinthine structure of the cave would not allow the player to follow the story correctly if each object were placed in its corresponding room.

1.2. The Ambient

The game is set in a labyrinthic, dark cave where the tunnels are very narrow. To light the path up, the player needs to find lanterns that are randomly positioned in the map. In the cave, that are also rooms of the player's house where the glowing items are placed. The rooms in the middle of the cave are created to suggest a more oniric experience and to guide the player towards the place of the items. The items in the rooms are glowing and each of them symbolize a part of the story of the player.

To create the map, we utilized two types of assets: the cave asset[1] and the house asset[2].

For the cave, we opted to construct a maze-like structure to deliberately disorient the player. Within some paths, there are boulders that obstruct movement but allow visibility beyond the blocked passage.

The strategically placed boulders ensure that the distance from each room is approximately the same. Additionally, dead ends are avoided to prevent the player from getting trapped between a rock and the pursuing monster.

Throughout the cave, various lanterns are placed, allowing the player to approach and ignite their own lantern. These lanterns are positioned

[1] https://assetstore.unity.com/packages/3d/environments/dungeons/mine-92461
[2] https://assetstore.unity.com/packages/3d/props/interior/free-house-interior-223416

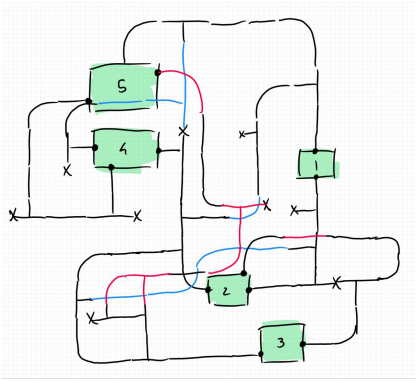

Figure 1: Map's drawing: red indicates an upper ground, blue indicates an underground and numbers indicates the house's room

randomly but are often found in corners or long corridors. Beyond helping the player, these lanterns are designed to illuminate the cave. For the second sprint, passive lanterns, used solely to brighten the cave, have been omitted.

We've incorporated a fog effect in the camera to limit visibility at a distance and reflection probes simulate the reflective impact of light on the stones.

During the map creation, we encountered challenges in connecting different parts of the cave. To address this, we plan to "fill the gaps" by adding rocks or placing boxes or beams in empty areas to conceal them. Another solution we implemented to mitigate this issue is the use of a dark skybox, reducing the visibility of fractures in the map.

To create the rooms, we based them on elements of the story, and each one represents a character in it. In each room, there is a panel representing the "owner" of the room, and it's possible to find the item that provides more information about the story. Each room is furnished in a meaningful way related to the narrative:

- father's room: a workspace to indicate how much the father was immersed in work, neglecting the family;
- daughter's room: features a bed symbolizing unity with the parents when stories were told by both;
- mother's room: a living room to indicate a moment of gathering with the family;

Figure 2: Father's room

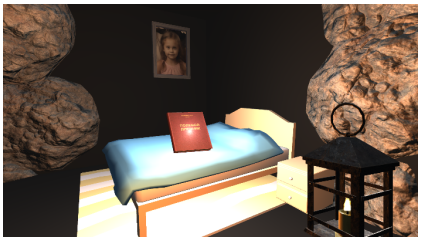

Figure 3: Daughter's room

- boss's room: a prison within the house, symbolizing the mafia's influence in the family's life;
- assassin's room: includes a hospital bed where the game's protagonist is currently located.

1.3. The Player

The player's visibility in the dark is provided, but it is extremely limited. The player has with him a candle that provides a small amount of light, allowing the player to see a short distance ahead. This is created so that the player needs the lantern and to not let him rush through the map. If the player runs for more than 2 seconds, the candle will extinguish, and he has to find a way to relight it. Additionally, he may unintentionally make noise by coughing or stumbling on rocks. The player's running speed is also slower than that of the monster: these mechanics are designed to make the player cautious about his movement. Inside the cave, there are several lanterns where the player can relight his candle. The location of these lanterns and the map are fixed and will never change. For the player to win, he must collect 5 objects in the map that are present in the house sections of the map. Every object represents a part of the story, and once collected, a letter or a diary page about the protagonists of the story will be revealed.

Figure 4: Wife's room

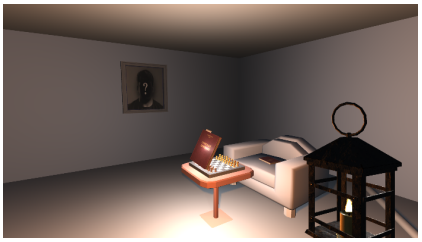

Figure 5: Boss' room

For the player's creation, we are currently using a Capsule object with a script (SC_FPSController.cs[3]) enabling basic movement using the W, A, S, and D keys on the keyboard for forward, backward, right, and left movements. The player's perspective is first-person, allowing observation of the surrounding environment using the mouse.

The player's lantern is an asset from the cave, childed to the camera so that it always remains in the player's view. To achieve a greater light effect, the Point Light of the player's lantern is positioned slightly ahead of the player's location to simulate an arm holding the lantern (given that the Capsule lacks arms).

The player can freely run and jump, with the caveat that running risks extinguishing the lantern and attracting the monster.

Currently, the script HideLightWithShift.cs[4] associated with the player turns off the lantern if the Shift button (for running) is pressed for more than 2 seconds, even if the player is stationary. This behavior needs adjustment.

To light up the lantern, the player needs to be in proximity to one of the active lanterns and press the E key. A prompt will appear to guide the player

[3]https://github.com/giacomoschiavo/WWAFA/blob/master/Assets/SC_FPSController.cs
[4]https://github.com/giacomoschiavo/WWAFA/blob/master/Assets/HideLightWithShift.cs

Figure 6: Tom's room

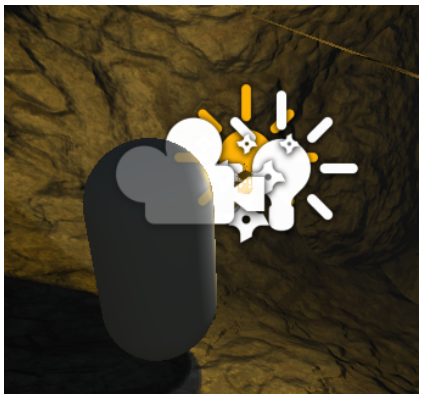

Figure 7: The player in 3rd person view

in this action. This functionality is implemented in ActivateLightOnPlayerNear.cs: the message and action become available when the player enters the activation range of the lantern defined by activationDistance, provided the lantern is currently off.

1.4. The Monster

The monster, Akumu, is a ferocious beast, whose sole purpose is to kill and tear apart any living being it encounters on its path. One of its main characteristics, is the fact that he is blind, so, in order to perceive its prey, it relies on his senses. Because of that, it detects the player only when his distance is inside a certain "range", and when in close proximity, it will attack. His usual behaviour then, will be to wander around randomly in the cave, and after detecting the player, it will immediately point in his direction, chasing him. The player still has a possibility to escape, by running faster than the beast.

The monster's model, has been imported from "Sketchfab", and all the

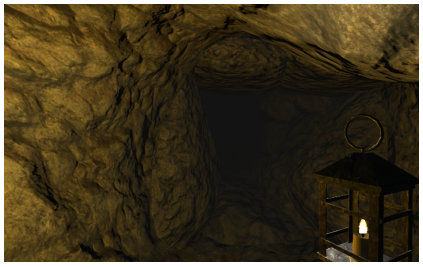

Figure 8: What the player see

Figure 9: Player structure

animations have been created and downloaded using "Mixamo", an online animation platform offering automated character rigging and a vast library of motion-capture animations.

2. Related work

There are several horror games that uses a similar approach for the AI of the enemies (that we are going to explain in the following paragraphs). Of course we are talking about more complex projects, that make use of sophisticated FSM and game mechanics. Here are two examples:

- Alien Isolation (2014): a survival horror game set in the Alien universe. The xenomorph enemy in the game uses a complex AI system with an FSM at its core. The alien can transition between states like patrolling, searching, and hunting the player based on their actions and the level of threat.

- Outlast (2013): a first-person survival horror game where players navigate a psychiatric hospital filled with hostile entities. The enemy AI, particularly the pursuing enemies like Chris Walker, uses an FSM to

Figure 10: Akumu, The Monster

transition between states such as patrolling, searching, and attacking based on the player's actions.

3. Proposal

3.1. AI Design

The AI of the monster is modeled using a finite state machine approach, a design that allows the enemy to switch between different states (Idle, wandering and Chase)based on specific conditions. It also makes use of the NavMeshAgent package, in order to permit the navigation of the agent in the scene.

3.2. Software architecture

The overall architecture is event-driven, where the behavior of the monster's AI is determined by various conditions and transitions between states. This type of architecture is common in game development, especially for simple AI systems.

The FSM is based on three states:

- Idle state: the enemy is in a passive (or resting) behaviour
- Wandering state: The enemy engages in controlled, random movements within a defined area. This state mimics exploratory behavior, creating a more lifelike and unpredictable presence.

- Chasing state: when the enemy detects a target, typically the player or another entity, it enters the Chasing state. This state activates pursuit behavior, where the enemy intelligently tracks and moves toward the detected target.

The monster general behaviour then (as we can see also in the graphical representation of the FSM, in figure 6), is the following:

- The monster starts in the walking state, randomly selecting destinations from the list and walking towards them.
- When the monster reach its destination, it enters the idle state, which last for a few seconds, in which he will randomly select another random destination. Then enters the walking state.
- While wandering, if the player is detected, the monster enter the chase state. It stops walking, triggers a chase animation, and runs towards the player using the NavMeshAgent.
- If the monster gets close enough to the player (within the catch distance), a jumpscare animation is triggered, and the player is considered caught: game over.

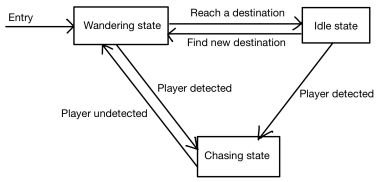

Figure 11: The Monster's animator

All the animation of the monster are handled by the "Unity Animator". The transitions between one animation and another, are triggerd when the monster change its state (passing from idle to walking for example). In Figure 6, we can see a screenshot of the monster's animator in Unity.

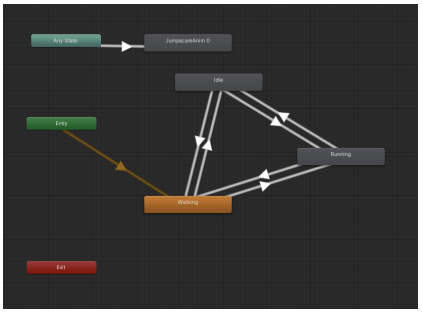

Figure 12: The Monster's animator

3.2.1. Screen Manager

To manage the story screens after an object has been collected, we've chosen to use a screen with fixed components while the text is dynamically handled. The displayed text always follows the same order to maintain narrative coherence.

Figure 13: Story screen

3.3. Next steps

in the following weeks, we will focus on two principle tasks. first of all we will work on defining some hiding spot, that when "activated" by the player, will immediately stop the chasing state of the monster, who will return in a wandering state.
Then, we plan to implement a long distance sound detection system, which will make the monster capable of locating the player's position when he makes some noise (for example when he picks up the objects), and promptly going to the indicated place. This will make the experience more challenging and fullfilling.

Based on the time left and level of complexity, we will then decide the next steps.

4. Simulations and Results

4.1. Methodology

4.2. Results

Demo for the 2nd sprint here. From this video it is possible to see the player, its mechanics and the working lanterns.

5. Conclusions and Future work

Give your conclusions and ideas for future work.

References