The Ring

By Fortoes Codrin

Game Type: 2D Adventure/Action

Score: Represented by the number of lives remaining at the end of the game.

System Design:

The character will be able to move forward, backward, right, and left using the (W, A, S, D) keys and will be able to attack with the left mouse click. During missions, the number of lives will appear dynamically in the top-left corner of the screen and will change if the player is attacked.

Context Design:

The main character's name is Akaad.

The action takes place in the Shurima Kingdom and its surroundings.

Akaad comes from a poor family with no means. However, the family's only wealth was the wife's ring, which some shady people stole one evening. Akaad must retrieve his wife's lost ring. The ring is a family heirloom passed down through generations. Enraged, he vows to take revenge. Nevertheless, he needs equipment to do this. Lacking money to purchase such equipment, he must complete tasks for the armor and weapon merchants in the village of Arkadia in the Shurima Kingdom. He must obtain his weapon (axe) and armor . After obtaining these items, he must fight the goblins in a cave, where he discovers her wife’s ring.

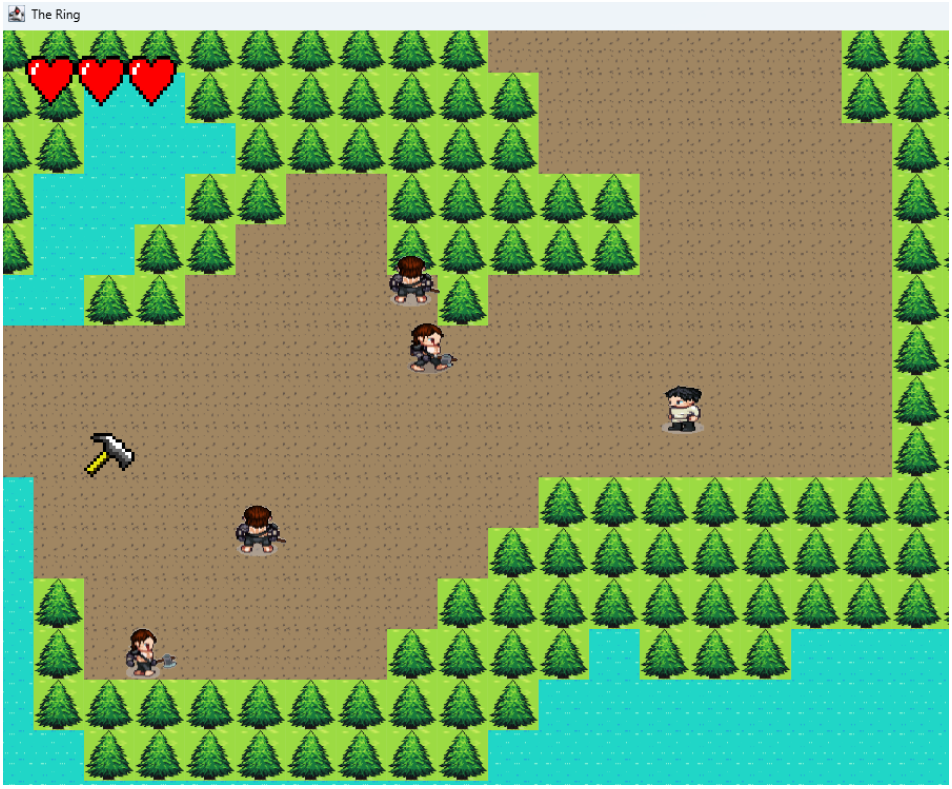
Content Design:

There will be 3 NPCs: the protagonist's wife, the weapon merchant, and the armor merchant. They have no abilities and do nothing except for the two merchants who can assign a mission to the player. No character in this game will have supernatural powers.

Level Design:

First Task: Defeating a pack of cursed wolves guarding a legendary axe. The wolves are in the forest area of Arkadia village, a place where everyone who ventures disappears without a trace. Without weapons, the player must lure the wolves. After the wolves take the bait, the player steals the axe, and as he tries to escape, the wolves realize it and try to attack him. The player must fight them, and if he kills them, the wolves will drop a claw that will help in a future quest.

Second Task: The armor merchant will tell the player he will craft special armor if paid sufficiently. The protagonist wants to give the claw in exchange for the armor, and the merchant agrees but offers rusted, degraded armor. Wanting to apply an ancient enchantment to the armor so it will never degrade, the player must complete a mission for the armor merchant. To do this, the protagonist must retrieve the merchant's hammer lost near a river. Upon reaching the river and searching for the hammer, the player is ambushed by thieves and must fight them to escape. The thieves have higher intelligence than the wolves, making this task more challenging. The thieves possess tools and armor, while the protagonist only has the axe and degraded armor.

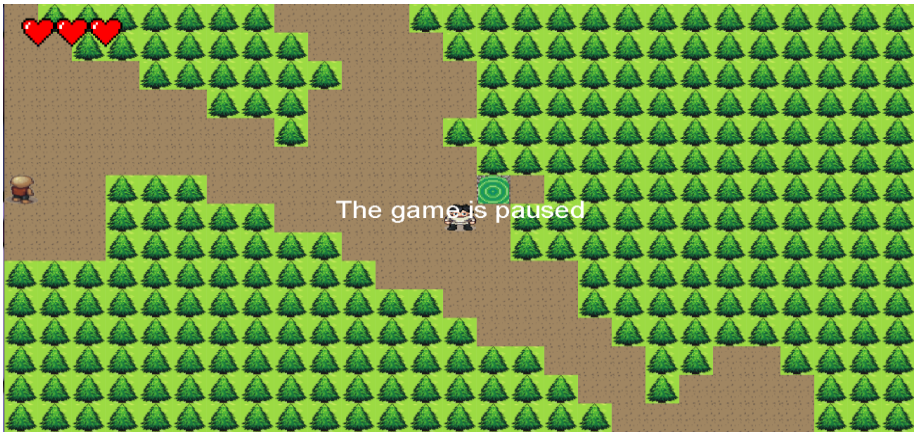


Third Task: Fighting goblins to recover the ring. The action takes place in a dungeon, and to reach the ring, the player must defeat the creatures in the cave. After clearing two rooms of monsters, the player completes the mission and retrieves the ring.

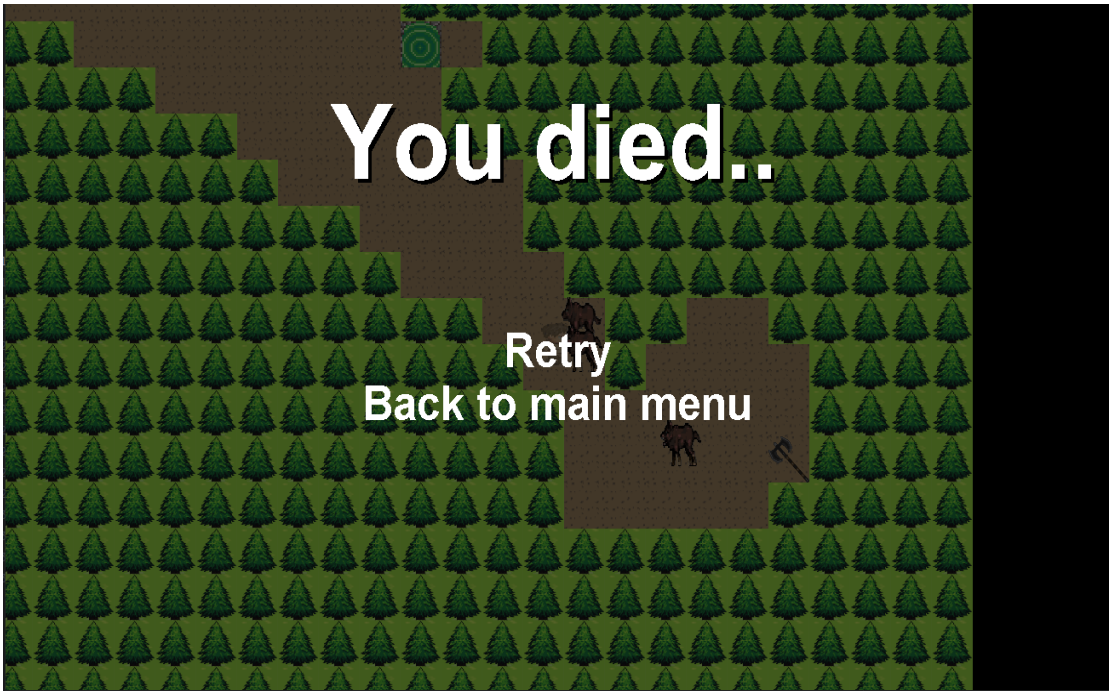
User Interface Design:

The user interface is simple, appropriate for a game of this genre. When the user steps into the game, they are greeted by a menu where they can start a new game or resume progress. If the user finishes the game, the endgame frame will appear on the screen. If the user dies, the text "You died" will appear along with a button that, when pressed, returns to the menu.

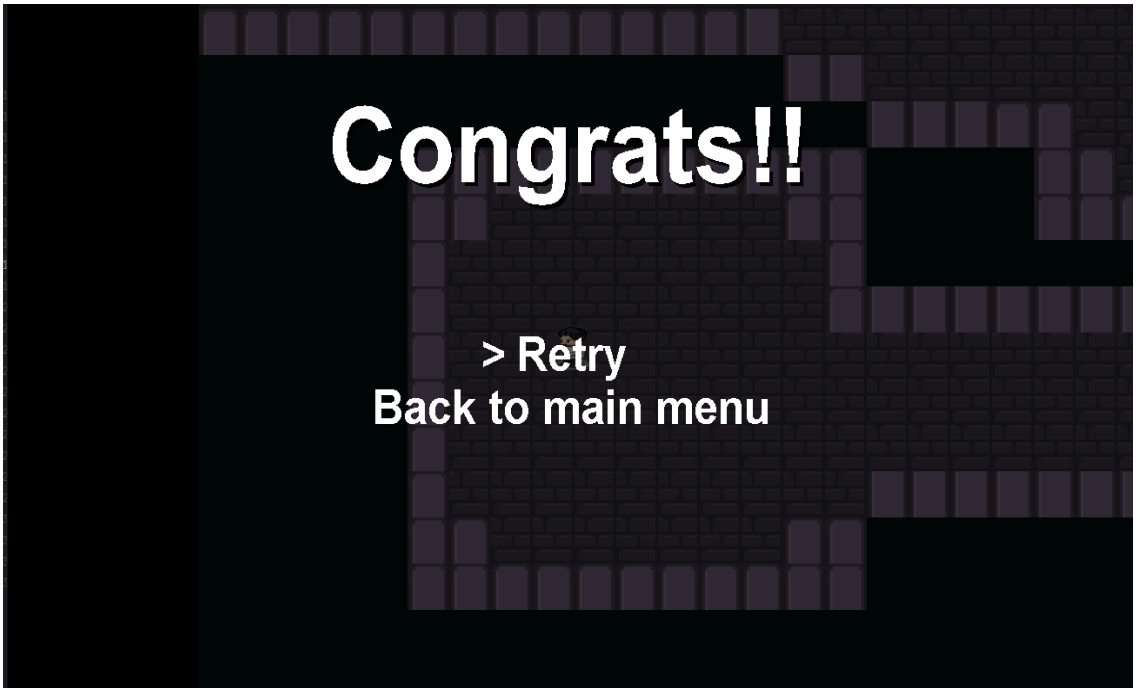
Pause State:



Death state:



Ending state:



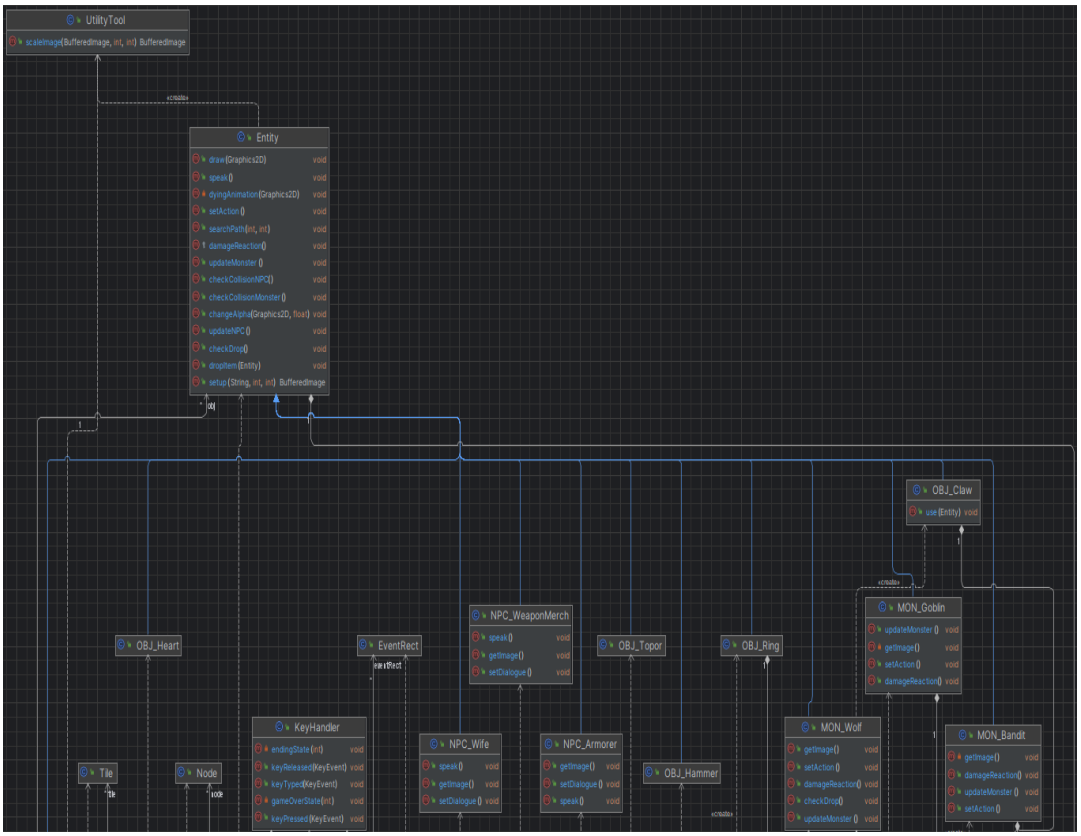
Database:

Table: tableTHERING

	Level	Health
	Filter	Filter
1	0	6
2	0	6
3	0	6
4	1	6
5	1	2
6	2	6
7	2	6
8	2	6
9	1	6
10	2	2
11	2	3
12	2	4
13	2	6
14	2	6
15	2	6
16	1	6
17	2	6
18	2	3
19	1	6
20	2	6
21	2	4

The database stores the player's level and life, modifying them based on the player's activity.

UML Diagram(part 1):



- **Entity:**

  Represents entities in the game, such as characters or movable objects. It has variables for managing the appearance and behavior of entities. Variables and methods are not specified in detail.

- **Tile:**

  Represents a piece (or tile) of a game. It has instance variables like image, collision, worldY, worldX, name, etc. The collision method is defined. There are methods for accessing and setting these variables.

- **Utility Tool:**

  A utility class providing general functionalities for image manipulation.

- **OBJ_Heart:**

  Represents a heart object in the game, used to give life or health to the player.

- **Node:**

  Used for pathfinding algorithms, representing a point in a network or map.

- **KeyHandler:**

Manages keyboard inputs, recording key presses and translating them into game actions.

- **NPC_WeaponMerch:**
  Represents an NPC giving the weapon quest to the player.
- **OBJ_Axe:**
  Represents an axe object in the game, used as a weapon.
- **OBJ_Ring:**
  Represents a ring object in the game, the endgame condition.
- **NPC_Wife:**
  Represents an NPC with the role of the wife in the game, which gives the main task.
- **NPC_Armorer:**
  Represents an NPC dealing with armors, giving the player the hammer quest.
- **OBJ_Hammer:**
  Represents a hammer object in the game, used as a quest object.
- **OBJ_Claw:**
  Represents a claw object in the game, used as a quest object.
- **MON_Goblin:**
  Represents a goblin monster, with specific behaviors and characteristics.
- **MON_Wolf:**
  Represents a wolf monster, with specific behaviors and characteristics.
- **MON_Bandit:**
  Represents a bandit monster, with specific behaviors and characteristics.

UML Diagram(part 2):

- **Main:**

  Contains the main method, which is the entry point of the application. The main method takes an array of strings (String[]) as an argument and does not return anything (void).

- **CollisionChecker:**

  Responsible for checking collisions in the game.

- **GamePanel:**

  Represents a game panel. It has variables for managing the game panel, such as screen dimensions, images, references to other game objects, etc. The draw method is defined to draw game elements on the panel. There are methods to check and update game objects.

- **Player:**

Represents the player in the game, including all possible actions and interactions, such as attacking monsters, interacting with NPCs, managing health, and collecting items.

- **UI:**

  Manages the graphical user interface, including displaying messages, dialog screens, game end screens, the player's health, and other UI elements.

- **Sound:**

  Manages sound in the game, including playing, stopping, looping, and setting audio files.

- **Pathfinder:**

  Manages pathfinding, including setting and resetting nodes, opening nodes, and tracking paths.

- **TileManager:**

  Manages game tiles, loading maps, and drawing them.

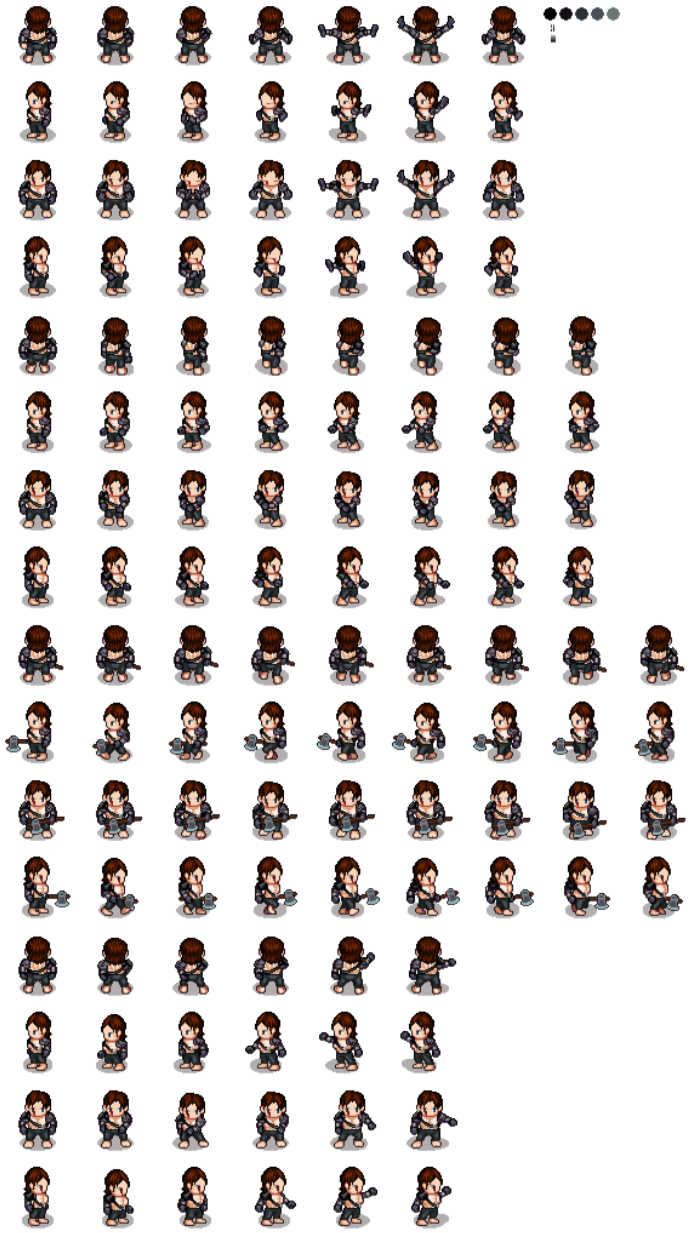
- **EventHandler:**

  Manages game events, including teleportation and interaction with game events.

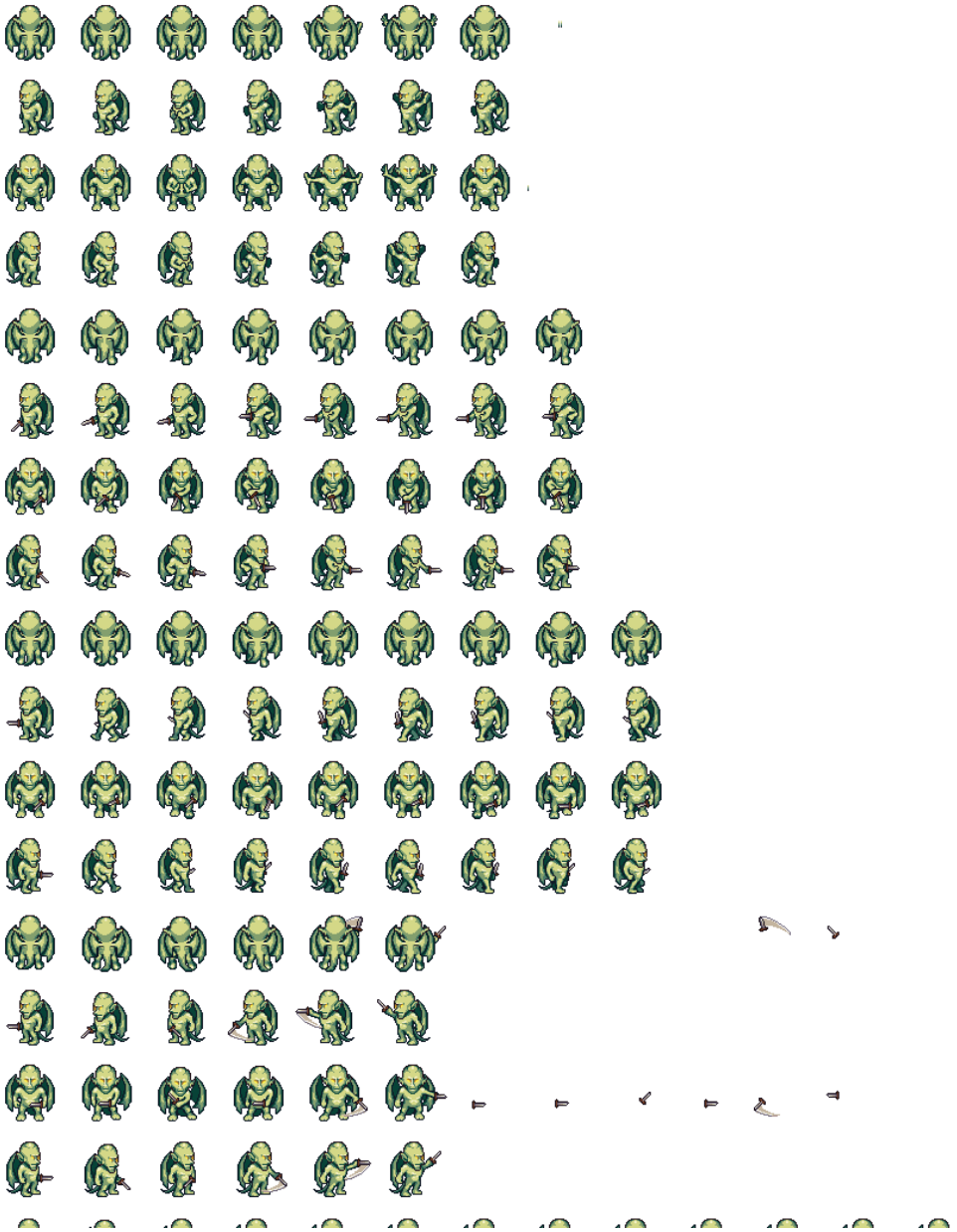
- **AssetSetter:**

  Initial setup of objects, NPCs, and monsters in the game.

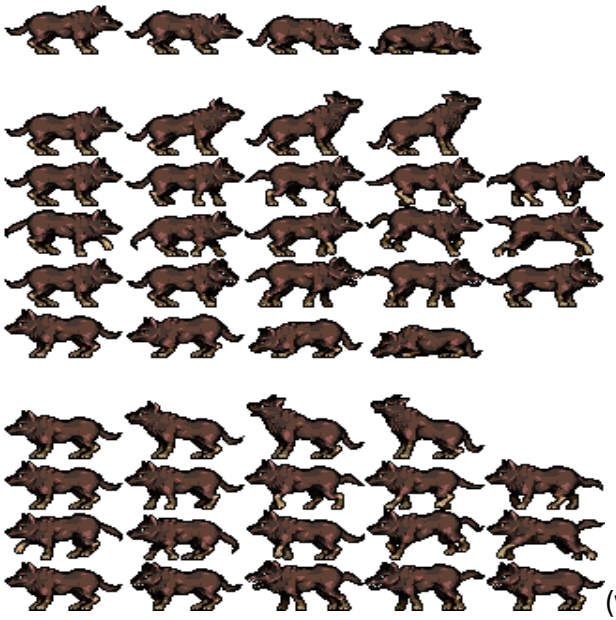
Game Sprites to be used (examples):



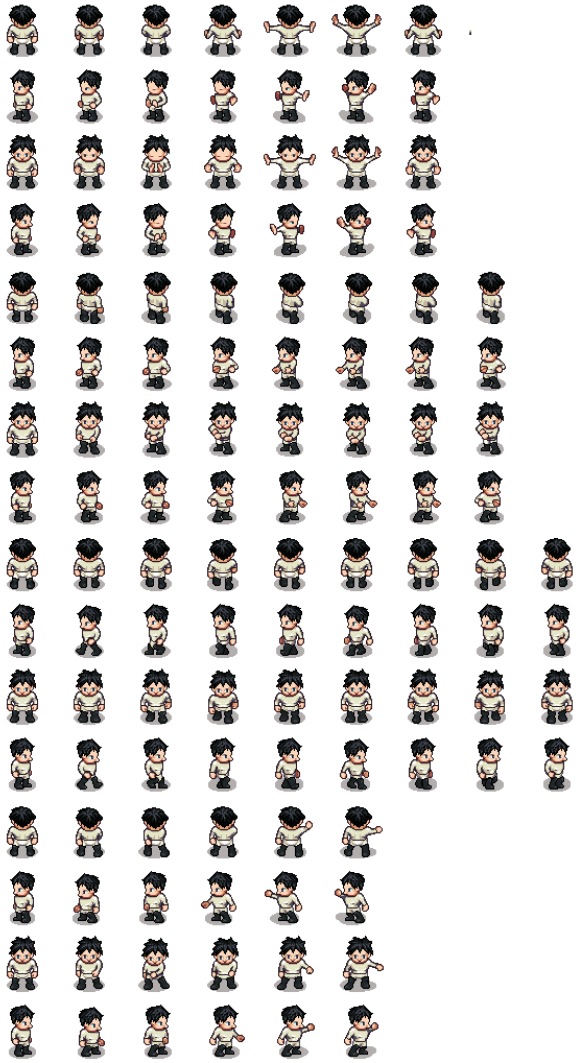
(bandit)



(goblin)



(**wolves**)



(**the main character**)

Bibliography:

1.https://opengameart.org

2. https://sanderfrenken.github.io/Universal-LPC-Spritesheet-Character-Generator/#?body=Body_color_light&head=Human_male_light

3. https://wepik.com/ai-generate