Semestral Project

Mini-game

Tópicos de Programação para Jogos

https://github.com/detiuaveiro/tpj-102536-102778

16/12/2024

universidade de aveiro
theoria poiesis praxis

Leonardo Almeida 102536
Pedro Rodrigues 102778

Game

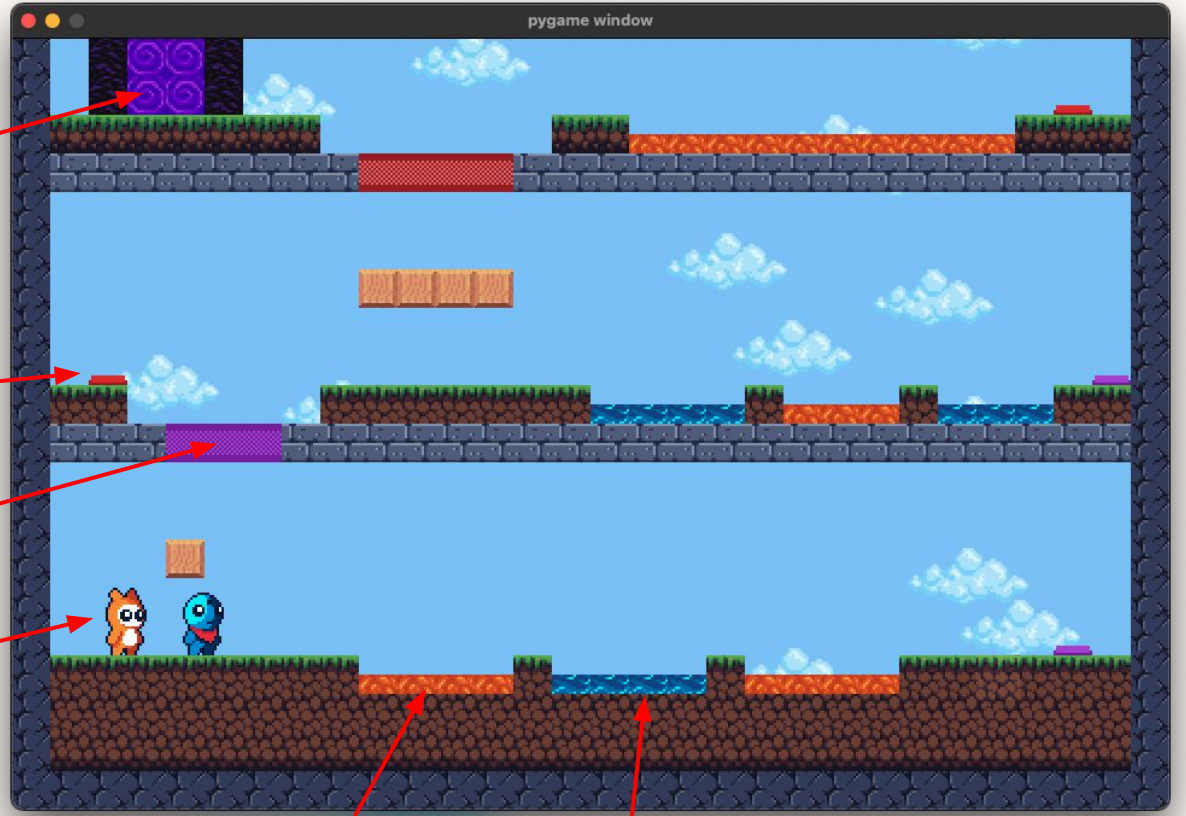
Original game: Fireboy and Watergirl

How it works:

- Two players: Fireboy and Watergirl
- Several levels, each one with an exit
- Levels have obstacles, mechanisms and deadly fluids
- A mechanism has one or multiple triggers that opens a barrier
- **The goal is to reach the exit by helping each other**
- **If a player dies, the level restarts for both players**

Original Game

Game



Exit
Players press "use" key to exit

Mechanism

Trigger
If pressed, opens barrier

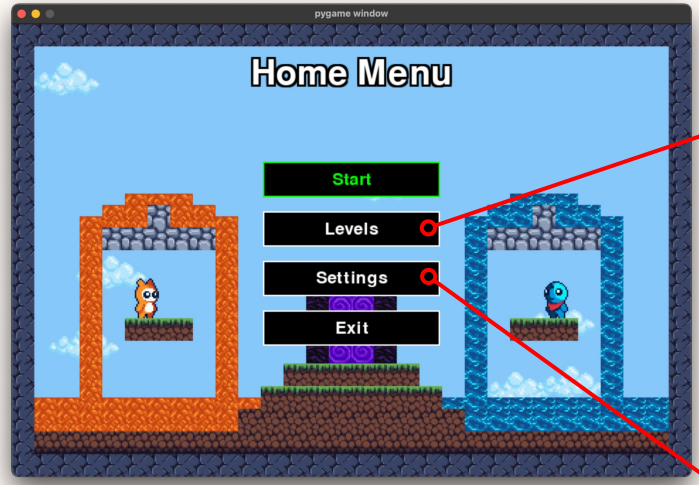
Barrier

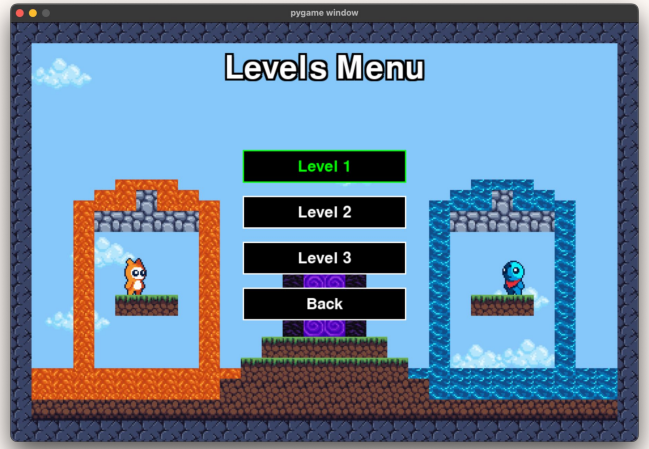
Players
Can walk, run, jump and "use"

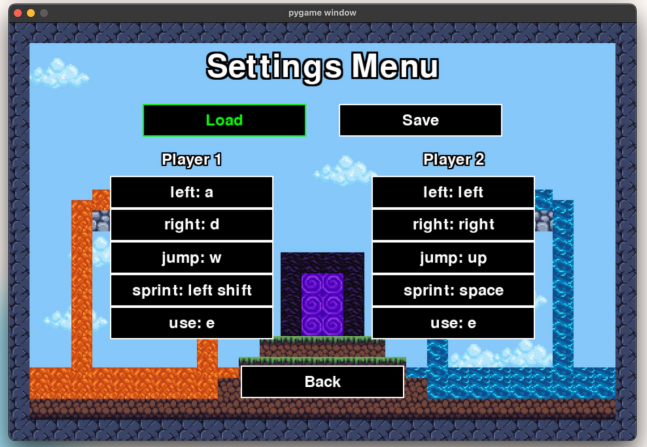
Lava
Only Fireboy can resist

Water
Only Watergirl can resist

Game







- 3 levels developed (+ easter egg)
- Users can change key binds
- Keys configuration can be saved and loaded from a json file
- Ability to pause, resume and change levels

Game Architecture

Modules:

- assets:
  Textures used by the sprites.
- maps:
  Bytecode for each level, alongside a config file.
- utils:
  General classes that use patterns and can be extended for any game.
- game:
  Constants and implementation of the main game logic.
- entities:
  Classes that interact with game events.
- sprites:
  Pygame sprite representation for entities that need it.

Patterns

Entity Locator:

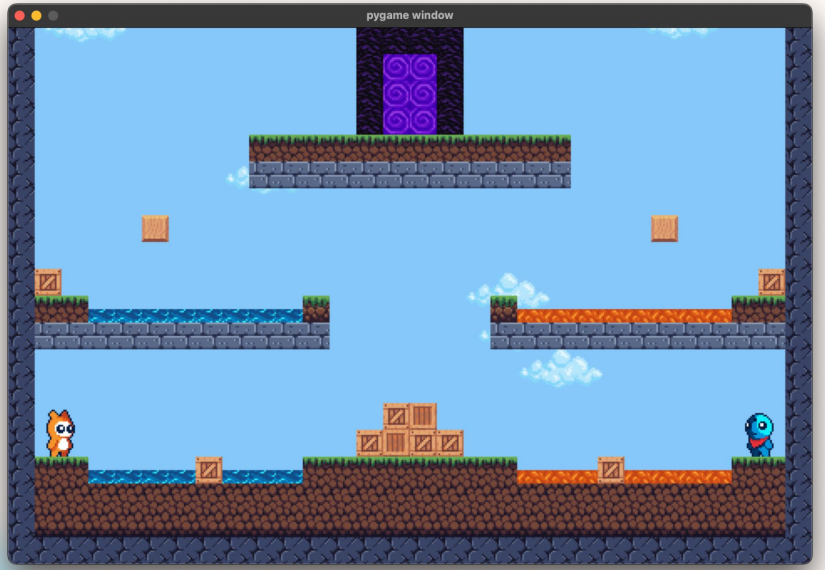
- Responsible to store, filter and retrieve entities.

Singleton:

- Used by the Locator, Sound and Event Queue, ensuring all calls occur to the same instance.

Event Queue:

- Receives events with optional arguments, allows Observers to register callbacks for these events and returns them to the Subject.



Patterns

Component:

- The Game has the Menu and the Level Manager, which are entities that communicate by events. This way they can be easily replaced.

Observer and Subject:

- The Observer is responsible to register callbacks for events and the Subject to get those events and notify by executing the callbacks with the optional arguments.

Game loop and double buffer:

- The subject class also runs the game loop, processing the input, updating the game and rendering it. The rendering is done with double buffer.

Patterns

Subclass:

- The Entity class extends Observer, all other entities extend Entity and Game extends Subject. Classes implemented in the utils module can be used to develop another game.

Collisions:

- The Game class detects collisions, using the AABB algorithm implemented in Pygame.

Bytecode:

- To render the Map bytecode is used with a config file.

Command:

- The Character uses this pattern in a functional way to be able to change keybinds and associate a callback to it.

Patterns

Finite State Machine:

- The Character transitions are fully controlled by a FSM, so that unexpected situations can't occur. The Character defines his starting state and possible transitions, then it is updated according to his actions.

Type object:

- Entities have Sprites and/or FSMs associated to them using this pattern.

Flyweight:

- Images for map Tiles are loaded once and reused by several tiles.

References

Players sprite sheet: Craftpix

Map tileset: Piiixl

Map editor: Tiled

Sounds: Myinstants, Pixabay

Photo editing: Photopea

Original Game: Fireboy and Watergirl