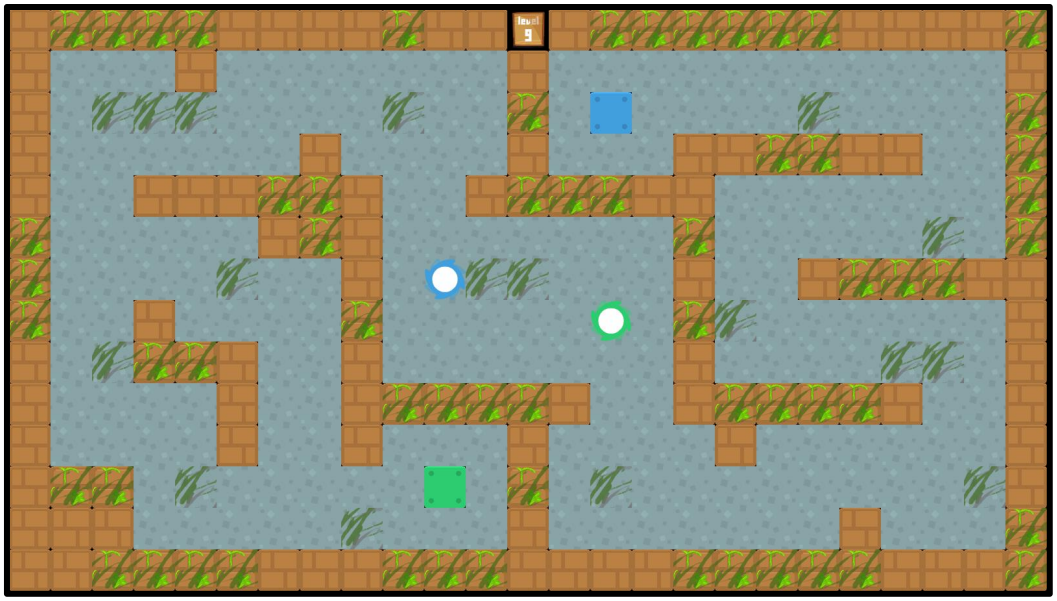
We Go

Arhip Constantin-Alexandru

Facultatea de Automatica si Calculatoare, Iasi

constantin-alexandru.arhip@student.tuiasi.ro

2020 - 2024



- **Gameplay:** *Jocul este single player, unde player-ul controelaza simultan 2 personaje cu scopul de a le ajuta pentru in a gasi iesirea din labirint. Gameplay-ul implica rezolvarea labirinturilor, insa pentru rezolvarea acestui obiectiv, jucatorul controleaza ambele personaje, miscarile lor petrecandu-se simultan, in aceeasi directie, astfel adaugand un pic de dificultate jocului, insa te poti folosii de peretii din cadrul nivelului, pentru "a te prinde" de ei, astfel asigurand miscarea doar a unui jucator. Totodata, pe langa acest lucru, rezolvarea labirinturilor trebuie sa se realizeze intr-un anumit timp specific fiecarui nivel. In cazul in care nu reusesti sa te incadrezi in acel timp, personajele tale vor fi teleportate inapoi la pozitia initiala a nivelului curent.*

- **Plot:** *Tu si amicul tau ati stat treji toata noaptea jucand "Wonderland Secret Worlds", un puzzle game. In timp ce asteptai pentru ati veni randul, ai adormit. Dupa multe ore de jucat, ai inceput sa visezi ca esti intr-un castel, similar cu cel din jocul la care te concentrai, intruchipandu-i pe Loof si Stinky.*

- **Characters:** ... (*ex:*
    - ***Loof*** *un explorator cu experienta, curajos, care a fost in numeroase aventuri si de fiecare data le-a terminat cu bine.*
    - ***Stinky*** *un explorator tanar, al carui model este Loof, el dorind sa fie exact ca el odata cu trecerea timpului, poate chiar mai bun.*
- **Mechanics (turns, game points, user interaction, keys):**
    - W/A/S/D pentru a misca caracterele in directia dorita;
    - ESC pentru a inchide jocul;
    - Butonul de START pentru a incepe jocul;
    - Butonul de TUTORIAL pentru a deschide tutorialul cat si o scurta prezentare a jocului;
    - Butonul de EXIT pentru a inchide jocul;
    - Iconita de MUZICA pentru a opri/porni muzica;
    - Iconita de EXIT pentru a iesi mai repede din joc atunci cand suntem in nivel;

La evaluare se vor avea in vedere urmatoarele:

#	Criteriu	Realizat
1	Abstractizare	10
2	Încapsulare	10
3	Moștenire (ierarhie de grad 3 minim)	10
4	Polimorfism	10
5	Interfețe (clase abstracte)	8
6	Gestionarea erorilor (excepții)	8
7	Salvarea sau încărcarea configurației jocului (Lucrul cu fișiere)	10
8	Număr de niveluri cu dificultate graduală (minim 3)	10

1. Abstractizare: Proprietatile pentru obiectele de tip SpriteComponent si Map.

```
private:
    TransformComponent* transform;
    SDL_Texture* texture;
    SDL_Rect srcRect, destRect;

    bool animated = false;
    int frames = 0;
    int speed = 100;
```

```
private:
    std::string texID;
    float mapScale;
    int tileSize;
    float scaledSize;
};
```

2.Incapsulare:

* clasa Map;

```
#pragma once

#include <string>

class Map
{
public:
    Map(std::string tID, float ms, float ts);
    ~Map();

    void LoadNewMap(std::string path, int sizeX, int sizeY);
    void LoadMap(std::string path, int sizeX, int sizeY);
    void AddTile(int srcX, int srcY, int xpos, int ypos);

private:
    std::string texID;
    float mapScale;
    int tileSize;
    float scaledSize;
};
```

* clasa Animation

```
#pragma once

class  Animation
{
    public:
        int index;
        int frames;
        int speed;

        Animation() {}
        Animation(int i, int f, int s)
        {
            index = i;
            frames = f;
            speed = s;
        }
};
```

* clasa Assets

```cpp
#pragma once

#include <map>
#include <string>
#include "Texture.h"
#include "Vector2D.h"
#include "ECS.h"

class AssetManager
{
public:
    AssetManager(Manager* man);
    ~AssetManager();

    void AddTexture(std::string id, const char* path);
    SDL_Texture* GetTexture(std::string id);

private:

    Manager* manager;
    std::map<std::string, SDL_Texture*> Textures;
};
```

* etc…

3. Moștenire (ierarhie de grad 3 minim):

In clasa Map, atunci cand generam harta, ne folosim de Component.h pentru a face legatura cu celelalte clase, respectiv TileComponent care preia tot cu ajutorul Component.h , care face legatura spre TextureManager.h, de unde preluam metoda LoadTexture pe care o folosim pentru a adauga textura hartii.

4. Polimorfism: Clasa TileComponent care preia metodele din clasa Texture.

```cpp
class TileComponent : public Component
{
public:

    SDL_Texture* texture;
    SDL_Rect srcRect, destRect;

    TileComponent() = default;

    ~TileComponent()
    {
        SDL_DestroyTexture(texture);
    }

    TileComponent(int srcX, int srcY, int xpos, int ypos, int tsize, int tscale, std::string id)
    {
        texture = Game::assets->GetTexture(id);

        srcRect.x = srcX;
        srcRect.y = srcY;
        srcRect.h = srcRect.w = tsize;

        destRect.x = xpos;
        destRect.y = ypos;
        destRect.h = destRect.w = tsize * tscale;
    }

    void draw() override
    {
        TextureManager::Draw(texture, srcRect, destRect, SDL_FLIP_NONE);
    }

    void setPos(int x, int y)
    {
        srcRect.x = x * 64;
        srcRect.y = y * 64;
    }
};
```

5. Interfete(clase abstracte) ~ Nu e virtuala pura

```
class Component
{
public:
    Entity* entity;

    virtual void init() {}
    virtual void update() {}
    virtual void draw() {}

    virtual ~Component() {}
};
```

6. Gestionarea erorilor (excepții) - Pentru crearea ferestrii, pentru muzica.

```
try
{
    if (window == 0)
        throw "Error creating window.";

    std::cout << "Window created.\n";
}
catch (const std::exception& e)
{
    std::cout << e.what();
}
```

```
try
{
    if (sample == 0)
        throw "Error getting music.";
}
catch (const std::exception& e)
{
    std::cout << e.what();
}
```

```
try
{
    mapFile.open(path);
}
catch (const std::exception& e)
{
    std::cerr << e.what() << '\n';
}
```

7. Salvarea sau încărcarea configurației jocului (Lucrul cu fișiere): In cadrul fisierului Map.h, realizam citirea hartii din fisier.

```
    for (int x = 0; x < sizeX; ++x)
    {
        mapFile.get(tilex);
        srcY = atoi(&tilex) * tileSize;
        mapFile.get(tiley);
        srcX = atoi(&tiley) * tileSize;

        tiles[counter]->getComponent<TileComponent>().setPos(atoi(&tiley), atoi(&tilex));
        counter++;

        if (tilex == '0' && tiley == '4')
        {
            auto& pa(manager.addEntity());
            pa.addComponent<TransformComponent>(x * scaledSize, y * scaledSize, tileSize, tileSize, mapScale);
            pa.addComponent<ColliderComponent>("portalAlbastru");
            pa.addGroup(Game::groupPortal);
        }

        if (tilex == '0' && tiley == '7')
        {
            auto& pv(manager.addEntity());
            pv.addComponent<TransformComponent>(x * scaledSize, y * scaledSize, tileSize, tileSize, mapScale);
            pv.addComponent<ColliderComponent>("portalVerde");
            pv.addGroup(Game::groupPortal);
        }

        mapFile.ignore();
    }
}

mapFile.ignore();
```

8. Număr de niveluri cu dificultate graduală (minim 3): Trecerea de la un nivel la altul, cat si realizarea hartii specifice acelui nivel. (In total, proiectul realizat are 10 nivele).

```
if (level == 2)
{
    Game::Remove();
    Game::Timer = 0;
    map->LoadNewMap("Images/Levels/Lvl2/Lvl2.map", 25, 14);
    player1.getComponent<TransformComponent>().setPos(128, 120);
    player2.getComponent<TransformComponent>().setPos(1408, 120);
}
else if (level == 3)
{
    Game::Remove();
    Game::Timer = 0;
    map->LoadNewMap("Images/Levels/Lvl3/Lvl3.map", 25, 14);
    player1.getComponent<TransformComponent>().setPos(640, 120);
    player2.getComponent<TransformComponent>().setPos(896, 120);
}
else if (level == 4)
{
    Game::Remove();
    Game::Timer = 0;
    map->LoadNewMap("Images/Levels/Lvl4/Lvl4.map", 25, 14);
    player1.getComponent<TransformComponent>().setPos(384, 120);
    player2.getComponent<TransformComponent>().setPos(1152, 640);
}
else if (level == 5)
{
    Game::Remove();
    Game::Timer = 0;
    map->LoadNewMap("Images/Levels/Lvl5/Lvl5.map", 25, 14);
    player1.getComponent<TransformComponent>().setPos(128, 704);
    player2.getComponent<TransformComponent>().setPos(1408, 120);
}
```

Surse:

- https://www.kenney.nl/assets?q=2d = Texturi pentru joc
- http://lazyfoo.net/tutorials/SDL/
- https://www.youtube.com/watch?v=44tO977slsU
- https://wiki.libsdl.org/
- https://www.youtube.com/watch?v=1KD4Ae0tX0g&list=PL-K0viiuJ2RctP5nlJlqmHGeh66-GOZR_
- https://parallelrealities.co.uk/tutorials/adventure/adventure4.php

Cod GitHub:

https://github.com/HipEx15/WeGo_OOP