Bartosz Błażewicz, Czapla Kamil, Józef Kasprzycki

Opis projektu

- Inspiracją naszego projektu była gra *The Binding of Isaac*

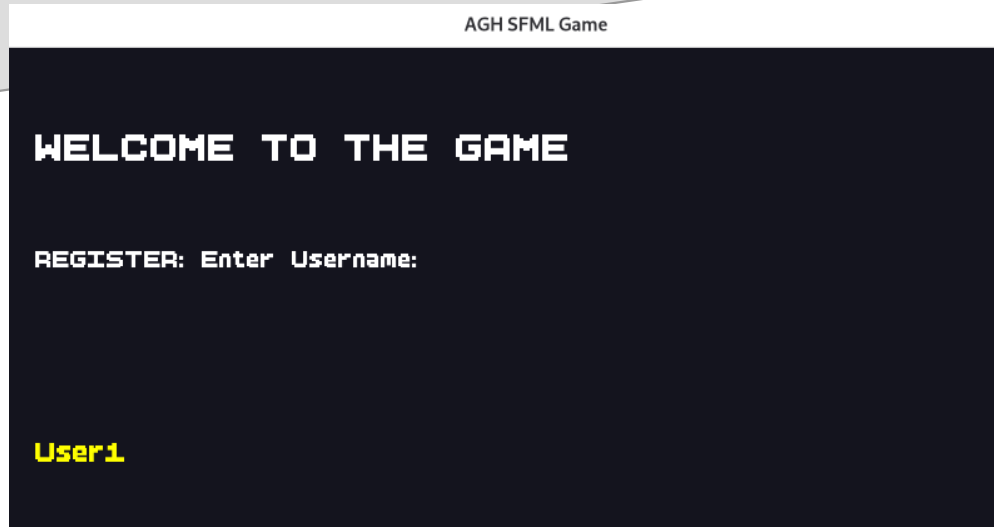
- Gra składa się z 4 poziomów: 3 zwykłych o wzrastającym stopniu trudności i jednego finałowego
- Na każdym poziomie na gracza czeka przeciwnik/czekają przeciwnicy do pokonania. Po pokonaniu przeciwników można przez specjalne drzwi przejść do kolejnego pokoju.
- W ostatnim poziomie czeka na gracza szef wszystkich przeciwników (boss) do pokonania.

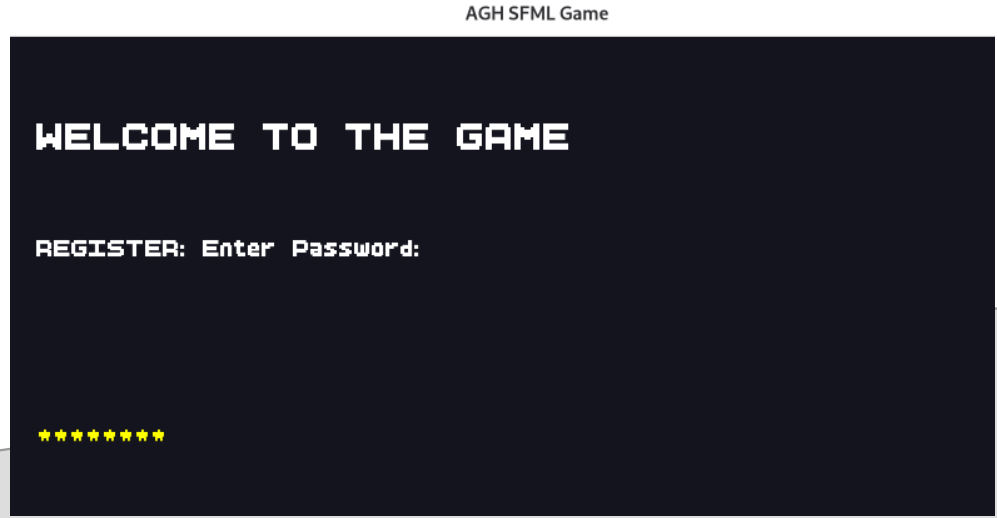
Wybrane funkcjonalności

Ekran rejestracji

```
case AuthState::RegInputUser:
    if (userManager.userExists(currentInput)) {
        errorText.setString("User already exists!");
        currentInput.clear();
    }
    else {
        tempUsername = currentInput;
        currentState = AuthState::RegInputPass;
        infoText.setString("REGISTER: Enter Password:");
        currentInput.clear();
    }
    break;

case AuthState::RegInputPass:
    tempPassword = currentInput;
    currentState = AuthState::RegConfirmPass;
    infoText.setString("REGISTER: Confirm Password:");
    currentInput.clear();
    break;
```

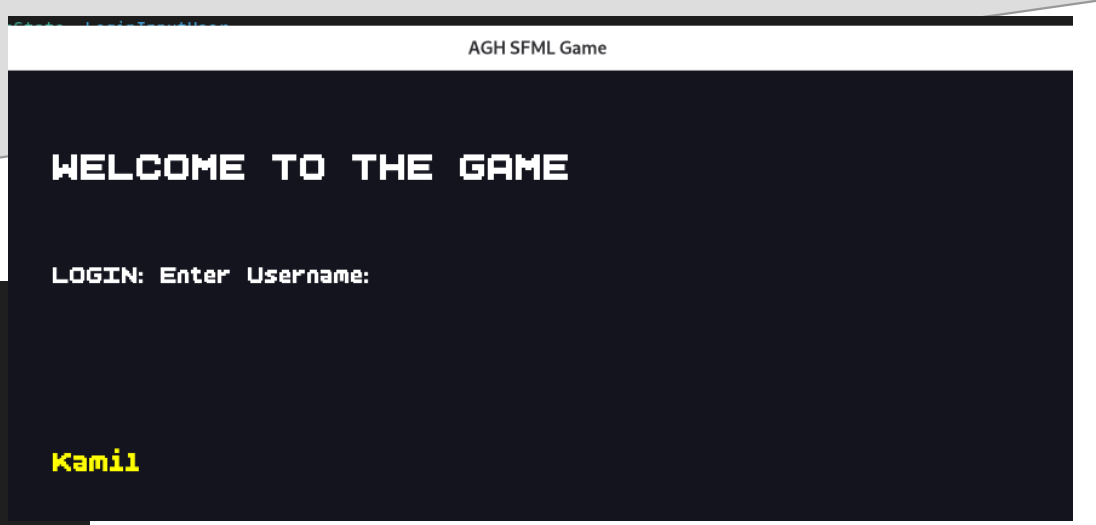
```
void AuthScreen::update(float delta) {
    // Maskowanie hasła gwiazdkami
    if (currentState == AuthState::LoginInputPass ||
        currentState == AuthState::RegInputPass ||
        currentState == AuthState::RegConfirmPass)
    {
        std::string masked(currentInput.length(), '*');
        inputText.setString(masked);
    }
    else {
        inputText.setString(currentInput);
    }
}
```

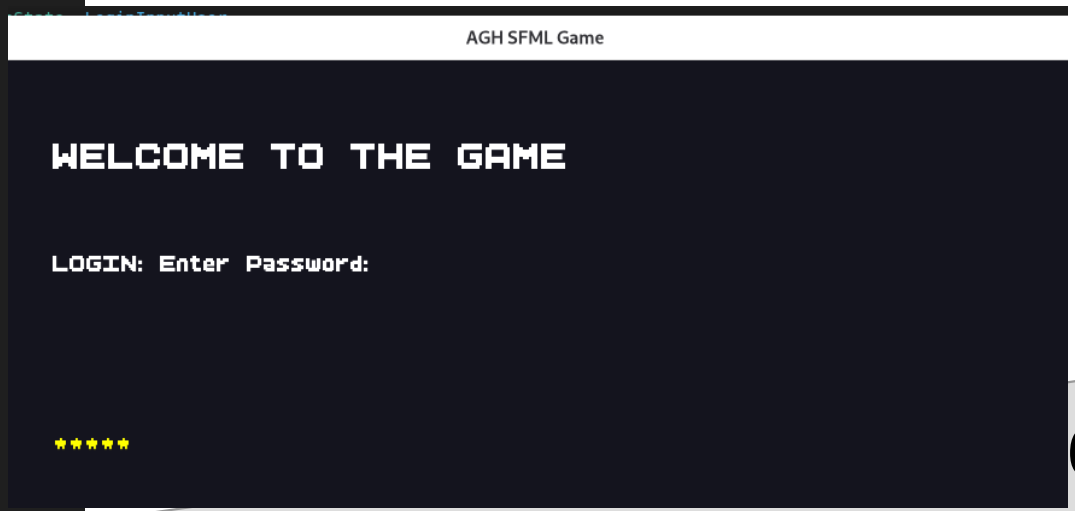




Ekran logowania

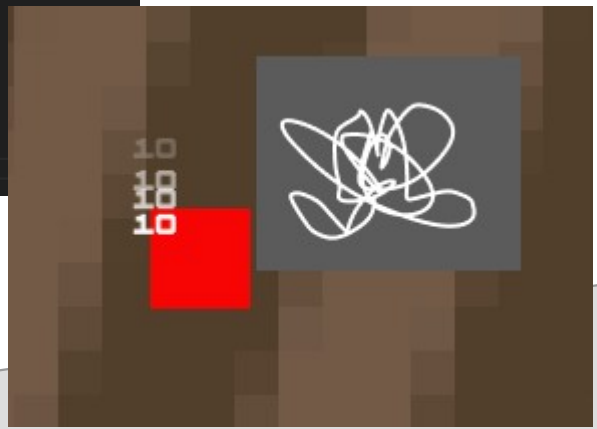
```
case AuthState::LoginInputUser:
    // Walidacja natychmiastowa: czy taki użytkownik w ogóle istnieje?
    if (userManager.userExists(currentInput)) {
        tempUsername = currentInput;
        currentState = AuthState::LoginInputPass;
        infoText.setString("LOGIN: Enter Password:");
        currentInput.clear();
    }
    else {
        errorText.setString("User does not exist!");
        // Zostajemy w tym samym stanie, użytkownik może poprawić login
    }
    break;

case AuthState::LoginInputPass:
    if (userManager.login(tempUsername, currentInput)) {
        // Logowanie udane - sprawdź czy to Admin
        GameScreen::setAdminMode(tempUsername == "Admin");
        finished = true;
    }
    else {
        errorText.setString("Invalid password! Press Enter to restart.");
        currentState = AuthState::Menu;
        infoText.setString("1. Create New Character\n2. Login\n3. Exit");
    }
    currentInput.clear();
    break;
```





Wyświetlanie powiadomień na ekranie

```cpp
FloatingText::FloatingText(const sf::Font& font, const std::string& msg, sf::Vector2f pos, sf::Color color)
    : lifetime(1.0f), maxLifetime(1.0f), velocity(0.f, -50.f) // Unoszenie się w górę
{
    text.setFont(font);
    text.setString(msg);
    text.setCharacterSize(20);
    text.setFillColor(color);
    text.setPosition(pos);

    // Wycentrowanie tekstu względem punktu
    sf::FloatRect bounds = text.getLocalBounds();
    text.setOrigin(bounds.left + bounds.width / 2.0f, bounds.top + bounds.height / 2.0f);
}

bool FloatingText::update(float delta) {
    lifetime -= delta;

    // Przesuwanie
    text.move(velocity * delta);

    // Zanikanie (Fade out)
    sf::Color c = text.getFillColor();
    c.a = static_cast<sf::Uint8>(255 * (lifetime / maxLifetime));
    text.setFillColor(c);

    return lifetime > 0.f;
}

void FloatingText::render(sf::RenderWindow& window) {
    window.draw(text);
}
```



Aktualne statystyki

```
// Wyświatlanie info w rogu ekranu:
// fps, czas, hp
leftCornerInfo.setString(
    std::to_string((int)(1/delta)) + " FPS\nTime: "
    + std::to_string((int)timeElapsedSec) + " sec."
    + "\nHP: " + std::to_string(player->getHP())
);
```

Tryb administratora – gracz nie doznaje obrażeń

```cpp
void PlayerBase::takeDamage(int dmg) {
    // Jeśli GodMode (Admin) lub tymczasowa nietykalność -> brak obrażeń
    if (godMode) return;
    if (invincibilityTimer > 0.f) return;
```

Szczegóły techniczne

- Szukanie folderu z danymi – dyrektywa #if defined (SYMBOL) – kompatybilność z Windowsem i Linuxem

```
#if defined(__linux__)
    : userManager("../data/users.json"), // Na linuxie wychodzi tylko jeden folder wyżej
#else
    : userManager("../../data/users.json"), // Na windowsie dwa foldery
#endif
```

- Zabezpieczenie przed wielokrotnym kompilowaniem – na początku każdego pliku nagłówkowego z klasami umieszczona dyrektywa #pragma once

```
1  #pragma once
2  #include "Screen.hpp"
3  #include "UserManager.hpp"
4  #include <SFML/Graphics.hpp>
5  #include <string>
```

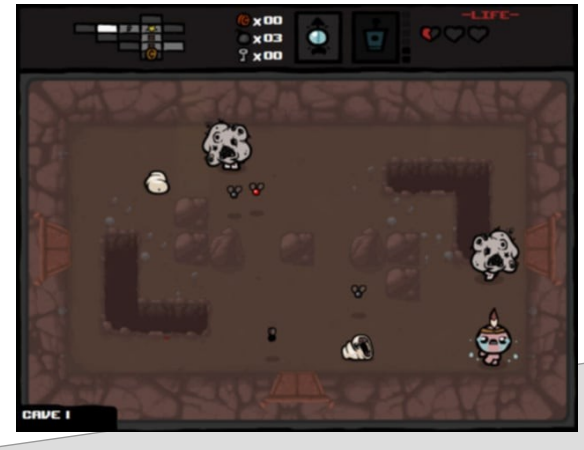
Kto co robił?

- Józef Kasprzycki – praca w kodzie (dużo pracy)
- Bartosz Błażewicz – pomysł, praca w kodzie (dużo pracy)
- Kamil Czapla – grafika, sprawozdanie, prezentacja

Bartosz Błażewicz, Czapla Kamil, Józef Kasprzycki

Opis projektu

- Inspiracją naszego projektu była gra *The Binding of Isaac*



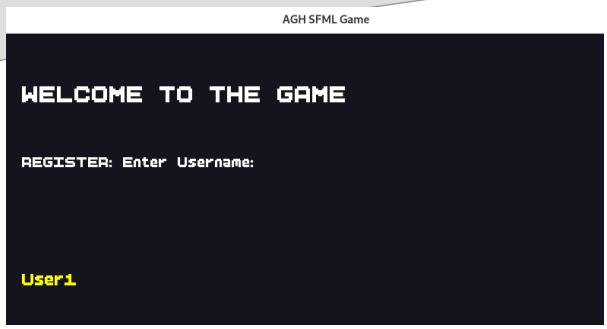
- Gra składa się z 4 poziomów: 3 zwykłych o wzrastającym stopniu trudności i jednego finałowego
- Na każdym poziomie na gracza czeka przeciwnik/czekają przeciwnicy do pokonania. Po pokonaniu przeciwników można przez specjalne drzwi przejść do kolejnego pokoju.
- W ostatnim poziomie czeka na gracza szef wszystkich przeciwników (boss) do pokonania.

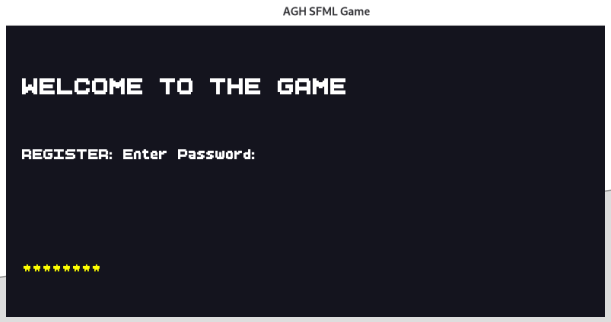
Wybrane funkcjonalności

Ekran rejestracji

```
case AuthState::RegInputUser:
    if (userManager.userExists(currentInput)) {
        errorText.setString("User already exists!");
        currentInput.clear();
    }
    else {
        tempUsername = currentInput;
        currentState = AuthState::RegInputPass;
        infoText.setString("REGISTER: Enter Password:");
        currentInput.clear();
    }
    break;

case AuthState::RegInputPass:
    tempPassword = currentInput;
    currentState = AuthState::RegConfirmPass;
    infoText.setString("REGISTER: Confirm Password:");
    currentInput.clear();
    break;
```

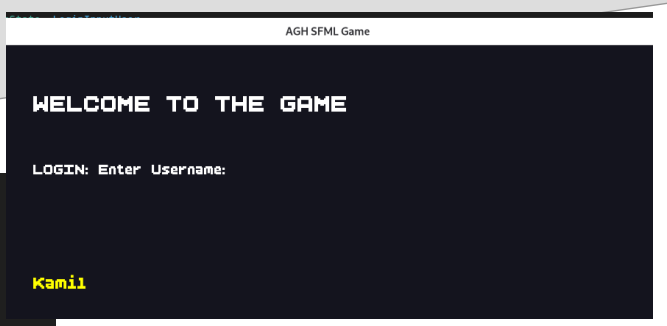
```
void AuthScreen::update(float delta) {
    // Maskowanie hasła gwiazdkami
    if (currentState == AuthState::LoginInputPass ||
        currentState == AuthState::RegInputPass ||
        currentState == AuthState::RegConfirmPass)
    {
        std::string masked(currentInput.length(), '*');
        inputText.setString(masked);
    }
    else {
        inputText.setString(currentInput);
    }
}
```

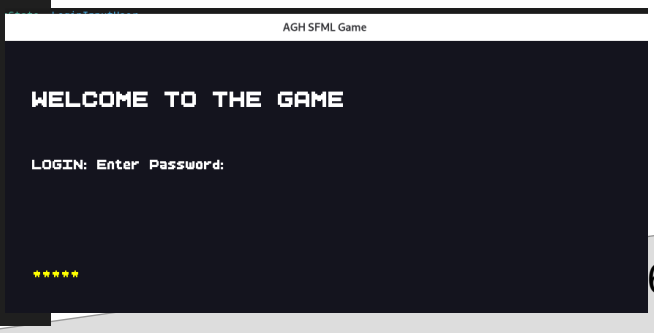




Ekran logowania

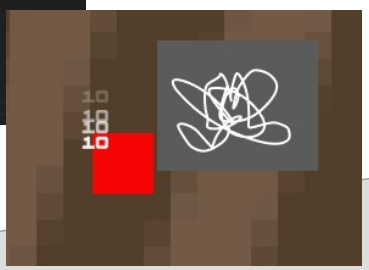
```
case AuthState::LoginInputUser:
    // Walidacja natychmiastowa: czy taki użytkownik w ogóle istnieje?
    if (userManager.userExists(currentInput)) {
        tempUsername = currentInput;
        currentState = AuthState::LoginInputPass;
        infoText.setString("LOGIN: Enter Password:");
        currentInput.clear();
    }
    else {
        errorText.setString("User does not exist!");
        // Zostajemy w tym samym stanie, użytkownik może poprawić login
    }
    break;

case AuthState::LoginInputPass:
    if (userManager.login(tempUsername, currentInput)) {
        // Logowanie udane - sprawdź czy to Admin
        GameScreen::setAdminMode(tempUsername == "Admin");
        finished = true;
    }
    else {
        errorText.setString("Invalid password! Press Enter to restart.");
        currentState = AuthState::Menu;
        infoText.setString("1. Create New Character\n2. Login\n3. Exit");
    }
    currentInput.clear();
    break;
```

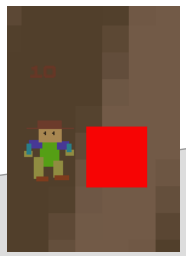




Wyświetlanie powiadomień na ekranie

```cpp
FloatingText::FloatingText(const sf::Font& font, const std::string& msg, sf::Vector2f pos, sf::Color color)
    : lifetime(1.0f), maxLifetime(1.0f), velocity(0.f, -50.f) // Unoszenie się w górę
{
    text.setFont(font);
    text.setString(msg);
    text.setCharacterSize(20);
    text.setFillColor(color);
    text.setPosition(pos);

    // Wycentrowanie tekstu względem punktu
    sf::FloatRect bounds = text.getLocalBounds();
    text.setOrigin(bounds.left + bounds.width / 2.0f, bounds.top + bounds.height / 2.0f);
}

bool FloatingText::update(float delta) {
    lifetime -= delta;

    // Przesuwanie
    text.move(velocity * delta);

    // Zanikanie (Fade out)
    sf::Color c = text.getFillColor();
    c.a = static_cast<sf::Uint8>(255 * (lifetime / maxLifetime));
    text.setFillColor(c);

    return lifetime > 0.f;
}

void FloatingText::render(sf::RenderWindow& window) {
    window.draw(text);
}
```





Aktualne statystyki

```
// Wyświatlanie info w rogu ekranu:
// fps, czas, hp
leftCornerInfo.setString(
    std::to_string((int)(1/delta)) + " FPS\nTime: "
    + std::to_string((int)timeElapsedSec) + " sec."
    + "\nHP: " + std::to_string(player->getHP())
);
```

Tryb administratora – gracz nie doznaje obrażeń

```
void PlayerBase::takeDamage(int dmg) {
    // Jeśli GodMode (Admin) lub tymczasowa nietykalność -> brak obrażeń
    if (godMode) return;
    if (invincibilityTimer > 0.f) return;
```

Szczegóły techniczne

- Szukanie folderu z danymi – dyrektywa #if defined (SYMBOL) – kompatybilność z Windowsem i Linuxem

```
#if defined(__linux__)
    : userManager("../data/users.json"), // Na linuxie wychodzi tylko jeden folder wyżej
#else
    : userManager("../../data/users.json"), // Na windowsie dwa foldery
#endif
```

- Zabezpieczenie przed wielokrotnym kompilowaniem – na początku każdego pliku nagłówkowego z klasami umieszczona dyrektywa #pragma once

```
1  #pragma once
2  #include "Screen.hpp"
3  #include "UserManager.hpp"
4  #include <SFML/Graphics.hpp>
5  #include <string>
```

Kto co robił?

- Józef Kasprzycki – praca w kodzie (dużo pracy)
- Bartosz Błażewicz – pomysł, praca w kodzie (dużo pracy)
- Kamil Czapla – grafika, sprawozdanie, prezentacja