Bartosz Błażewicz, Czapla Kamil, Józef Kasprzycki

Opis projektu

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

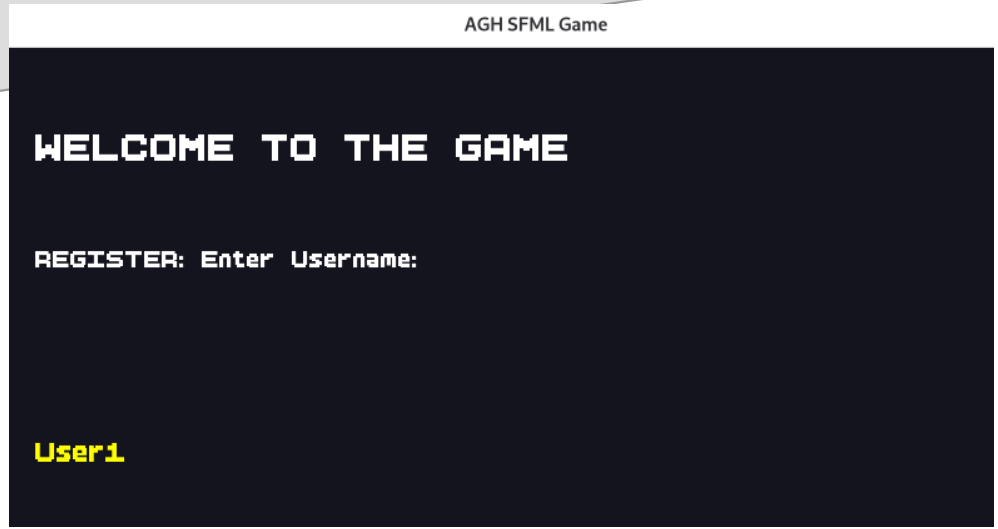
- Gra składa się z 4 poziomów: 3 zwykłych o wzrastającym stopniu trudności i jednego finałowego
- Na każdym poziomie na gracza czeka przeciwnik/czekają przeciwnicy do pokonania. Po pokonaniu przeciwników można przez specjalne drzwi przejść do kolejnego pokoju.
- W ostatnim poziomie czeka na gracza szef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

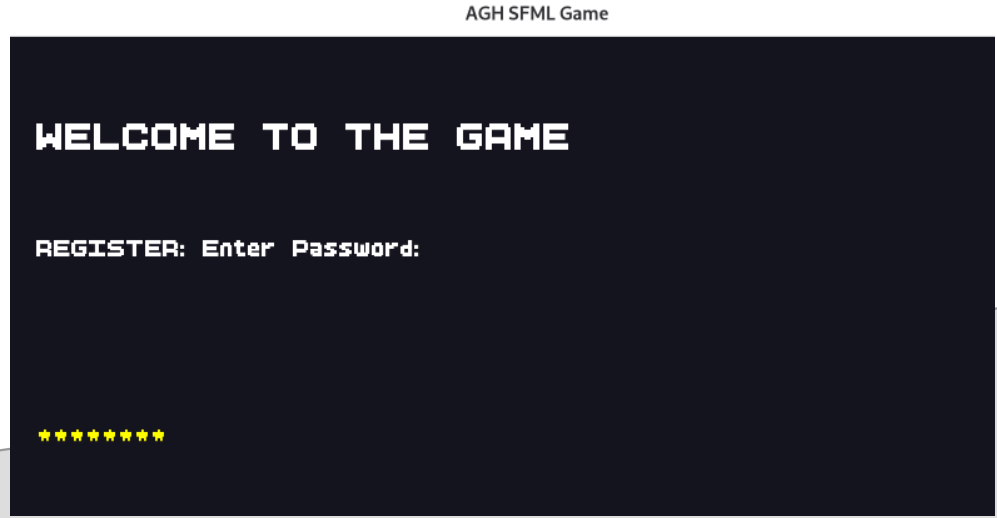
Wybrane funkcjonalności

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;
```

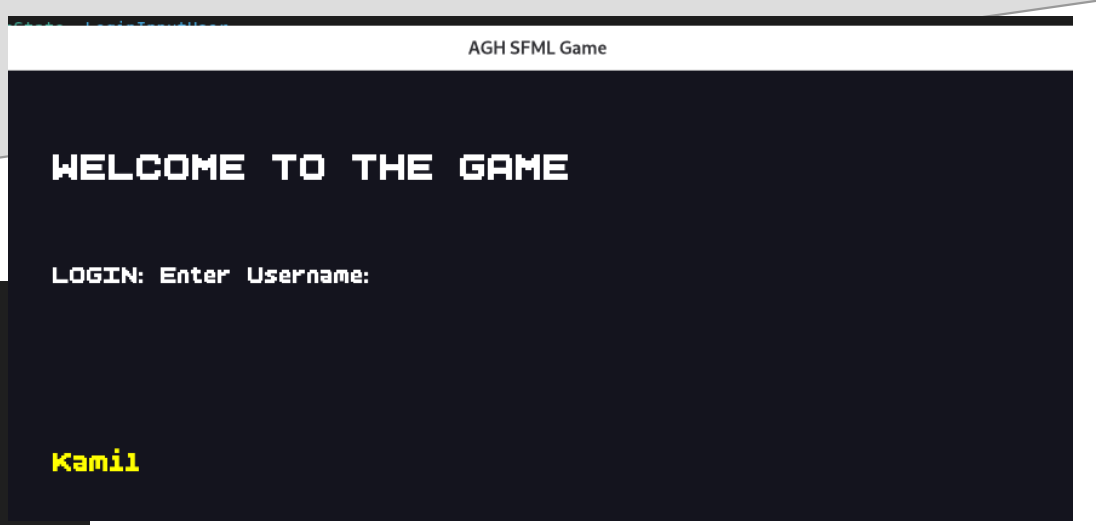
```
void AuthScreen::update(float delta) {
    // Maskowanie hasła gwiazdkami
    if (currentState == AuthState::LoginInputPass ||
        currentState == AuthState::RegInputPass ||
        currentState == AuthState::RegConfirmPass)
    {
        std::string masked(currentInput.length(), '*');
     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

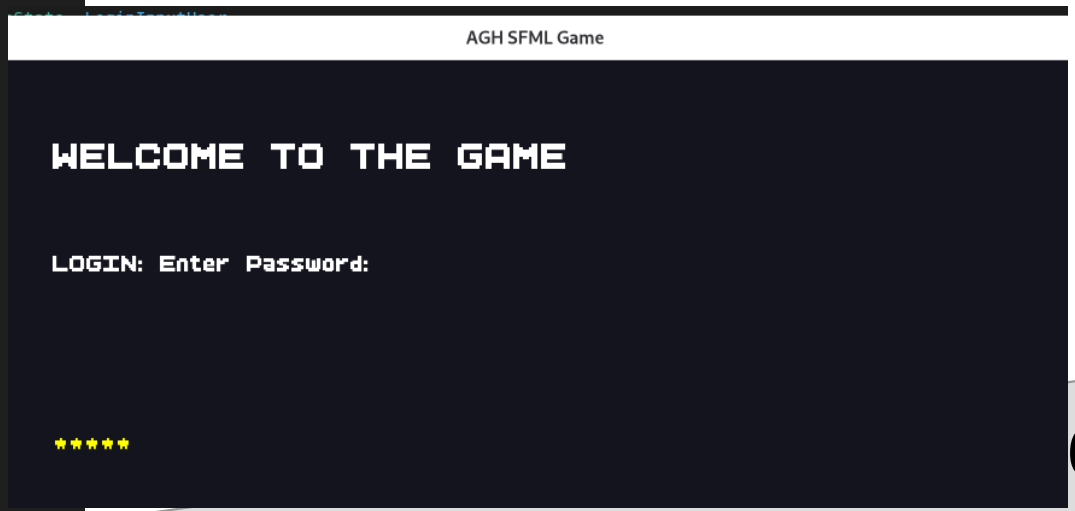




Ekran logowania

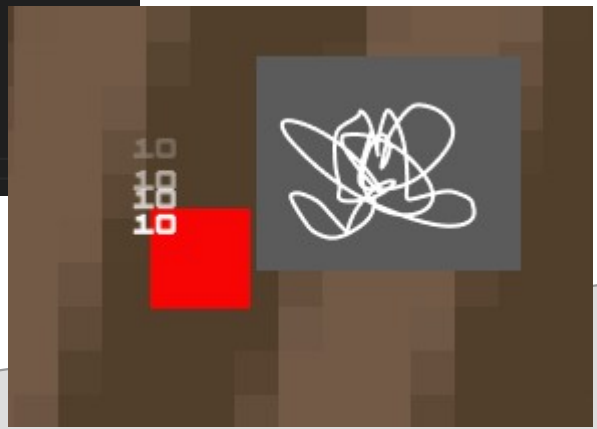
```
case AuthState::LoginInputUser:
    // Walidacja natychmiastowa: czy taki użytkownik w ogóle istnieje?
    if (userManager.userExists(currentInput)) {
        tempUsername = currentInput;
        currentState = AuthState::LoginInputPass;
        infoText.setString("LOGIN: Enter Password:");
        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

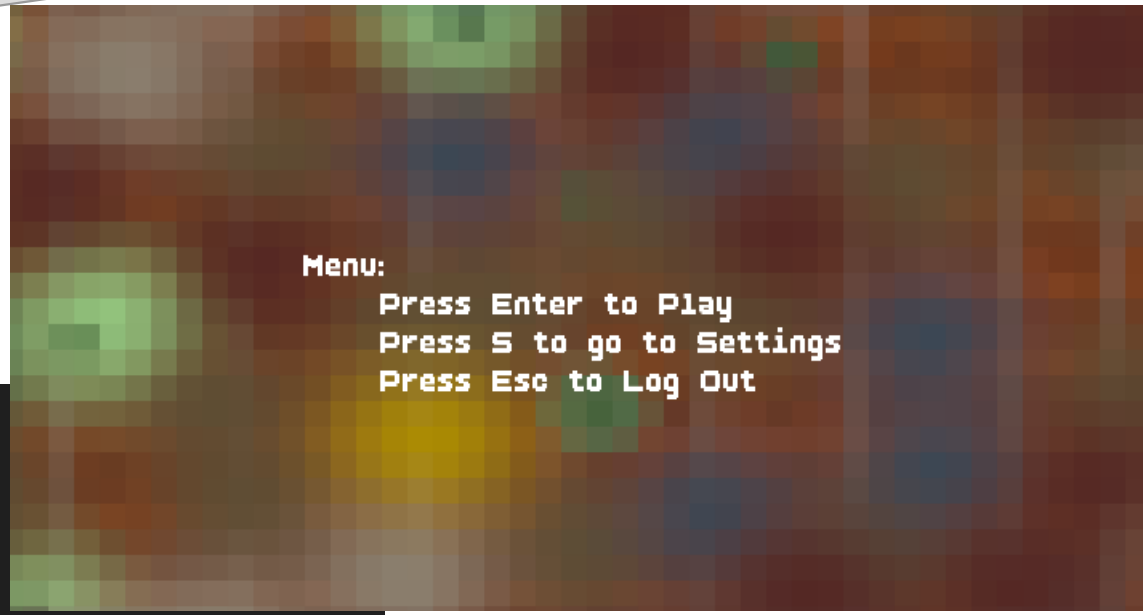
```cpp
FloatingText::FloatingText(const sf::Font& font, const std::string& msg, sf::Vector2f pos, sf::Color color)
    : lifetime(1.0f), maxLifetime(1.0f), velocity(0.f, -50.f) // Unoszenie się w górę
{
    text.setFont(font);
    text.setString(msg);
    text.setCharacterSize(20);
    text.setFillColor(color);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

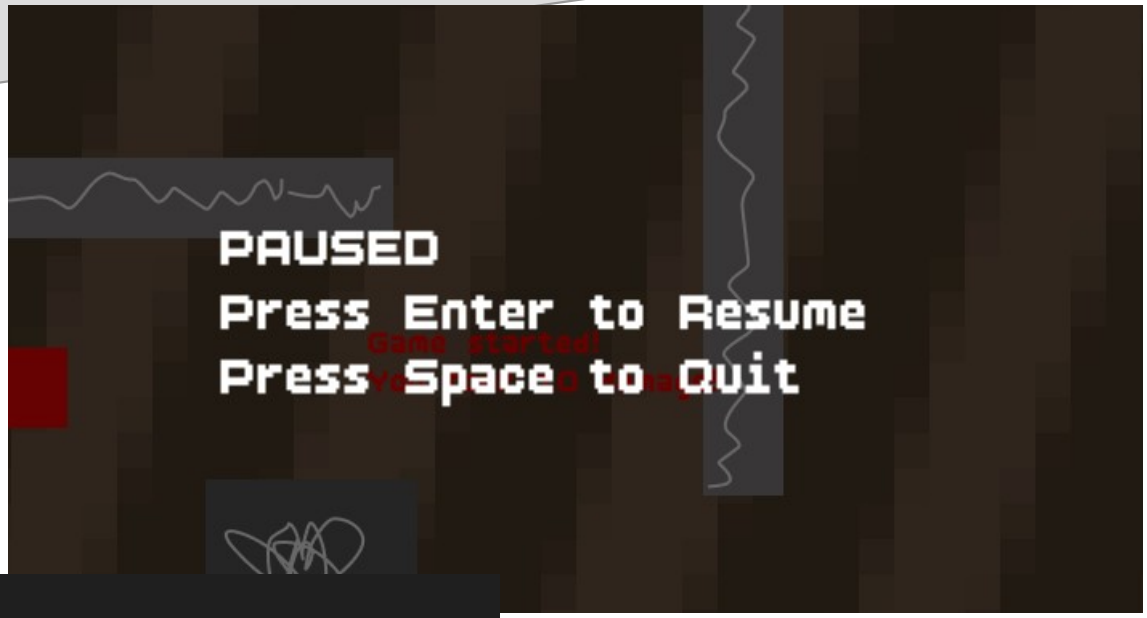
```
// Wyświatlanie info w rogu ekranu:
// fps, czas, hp
leftCornerInfo.setString(
    std::to_string((int)(1/delta)) + " FPS\nTime: "
    + std::to_string((int)timeElapsedSec) + " sec."
    + "\nHP: " + std::to_string(player->getHP())
);
```

Menu




```cpp
MenuScreen::MenuScreen() {
    // Zakładam, że TextureManager działa poprawnie
    try {
        backgroundTexture = TextureManager::get("../assets/menu_bg.png");
        backgroundSprite.setTexture(backgroundTexture);
    }
    catch (...) {
        // Fallback jeśli brak tła
    }

    if (!font.loadFromFile("../assets/font.ttf")) {
        // handle error
    }
    text.setFont(font);
    // Zmieniłem tekst "quit" na "Log Out", żeby pasował do akcji
    text.setString("Menu: \n\tPress Enter to Play\n\tPress S to go to Settings\n\tPress Esc to Log Out");
    text.setCharacterSize(24);
    text.setPosition(350, 250);
}
```

Pauza gry




```
// Tekst ekranu pauzy
pauseText.setFont(font);
pauseText.setString("PAUSED\nPress Enter to Resume\nPress Space to Quit");
pauseText.setCharacterSize(40);
pauseText.setFillColor(sf::Color::White);
// Centrowanie tekstu (dostosowane do d
sf::FloatRect textRect = pauseText.getLocalBounds();
pauseText.setOrigin(textRect.left + textRect.width / 2.0f, textRect.top + textRect.height / 2.0f);
pauseText.setPosition(500.f, 300.f);

// Maskownica ekranu w czasie pauzy
pauseOverlay.setSize(sf::Vector2f(1000.f, 600.f));
pauseOverlay.setFillColor(sf::Color(0, 0, 0, 150));
```

Ładowanie poziomu z pliku

```cpp
using json = nlohmann::json;

LevelData LevelLoader::loadFromFile(const std::string& path) {
    std::ifstream file(path);
    if (!file.is_open()) {
        throw std::runtime_error("Nie mozna otworzyc pliku poziomu: " + path);
    }

    json j;
    file >> j;

    LevelData level;
    level.name = j["name"];
    level.size.x = j["size"]["width"];
    level.size.y = j["size"]["height"];
    level.background = j["background"];

    level.playerStart.x = j["player"]["position"][0];
    level.playerStart.y = j["player"]["position"][1];
```

```cpp
// Przeszkody
for (const auto& o : j["obstacles"]) {
    ObstacleData obs;
    obs.bounds = sf::FloatRect(
        o["x"].get<float>(), o["y"].get<float>(),
        o["w"].get<float>(), o["h"].get<float>()
    );
    obs.texture_path = o["texture_path"].get<std::string>();
    level.obstacles.push_back(obs);
}

// Wrogowie
for (const auto& o : j["enemies"]) {
    EnemyData enemy;
    enemy.bounds = sf::FloatRect(
        o["x"].get<float>(), o["y"].get<float>(),
        o["w"].get<float>(), o["h"].get<float>()
    );
    enemy.type = o["type"].get<std::string>();
    level.enemies.push_back(enemy);
}
```

```cpp
48      // Drzwi (NOWE)
49      if (j.contains("doors")) {
50          for (const auto& d : j["doors"]) {
51              DoorData door;
52              door.bounds = sf::FloatRect(
53                  d["x"].get<float>(), d["y"].get<float>(),
54                  d["w"].get<float>(), d["h"].get<float>()
55              );
56              door.next_level_path = d["next_level"].get<std::string>();
57              level.doors.push_back(door);
58          }
59      }
60
61
62      return level;
63  }
```

Mechanika ruchu

```
97  void PlayerBase::applyMovementPhysics(float delta) {
98      // --- 1. FIZYKA RUCHU ---
99
100     // O Y
101     if (inputDirection.y < 0.f) { // W
102         if (speed_vector.y > -max_speed)
103             speed_vector.y -= max_speed * delta;
104     }
105     else if (inputDirection.y > 0.f) { // S
106         if (speed_vector.y < max_speed)
107             speed_vector.y += max_speed * delta;
108     }
109     else if (speed_vector.y > min_speed) {
110         speed_vector.y -= max_speed * delta;
111     }
112     else if (speed_vector.y < -min_speed) {
113         speed_vector.y += max_speed * delta;
114     }
115     else {
116         speed_vector.y = 0.f;
117     }
```

```
119     // O X
120     if (inputDirection.x < 0.f) { // A
121         if (speed_vector.x > -max_speed)
122             speed_vector.x -= max_speed * delta;
123     }
124     else if (inputDirection.x > 0.f) { // D
125         if (speed_vector.x < max_speed)
126             speed_vector.x += max_speed * delta;
127     }
128     else if (speed_vector.x > min_speed) {
129         speed_vector.x -= max_speed * delta;
130     }
131     else if (speed_vector.x < -min_speed) {
132         speed_vector.x += max_speed * delta;
133     }
134     else {
135         speed_vector.x = 0.f;
136     }
```

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>
```

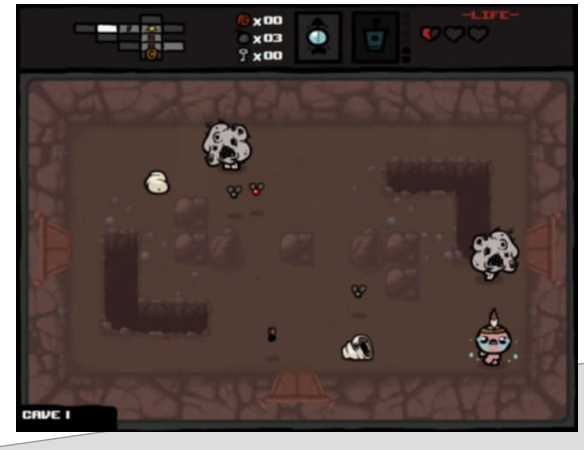
Kto co robił?

- 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*



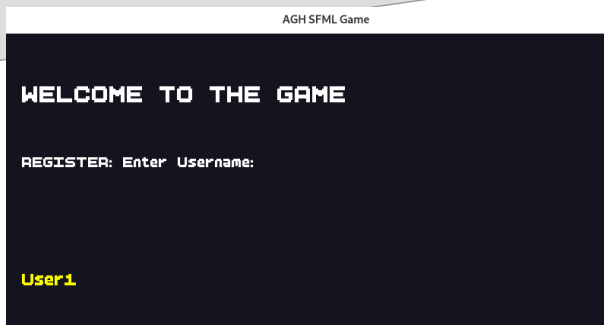
- Gra składa się z 4 poziomów: 3 zwykłych o wzrastającym stopniu trudności i jednego finałowego
- Na każdym poziomie na gracza czeka przeciwnik/czekają przeciwnicy do pokonania. Po pokonaniu przeciwników można przez specjalne drzwi przejść do kolejnego pokoju.
- W ostatnim poziomie czeka na gracza szef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

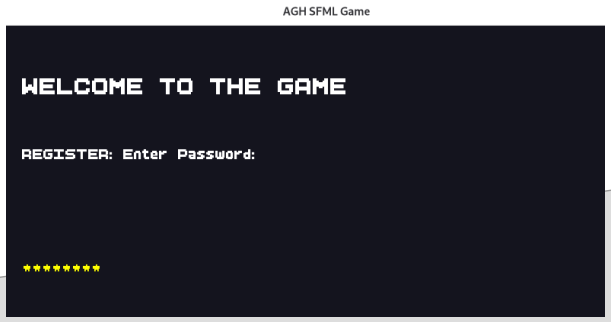
Wybrane funkcjonalności

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;
```

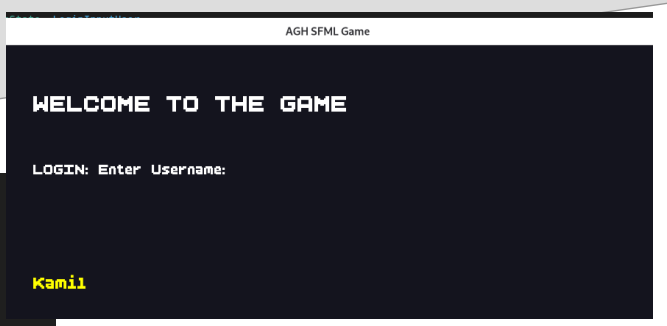
```
void AuthScreen::update(float delta) {
    // Maskowanie hasła gwiazdkami
    if (currentState == AuthState::LoginInputPass ||
        currentState == AuthState::RegInputPass ||
        currentState == AuthState::RegConfirmPass)
    {
        std::string masked(currentInput.length(), '*');
     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

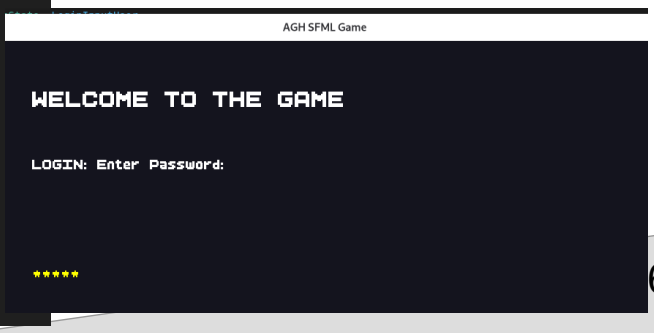




Ekran logowania

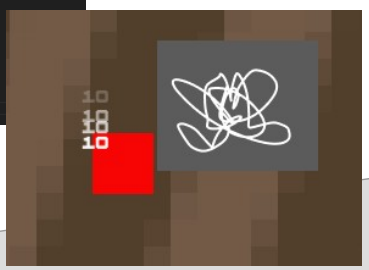
```
case AuthState::LoginInputUser:
    // Walidacja natychmiastowa: czy taki użytkownik w ogóle istnieje?
    if (userManager.userExists(currentInput)) {
        tempUsername = currentInput;
        currentState = AuthState::LoginInputPass;
        infoText.setString("LOGIN: Enter Password:");
        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;
```

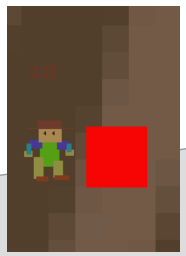




Wyświetlanie powiadomień na ekranie

```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);
}
```

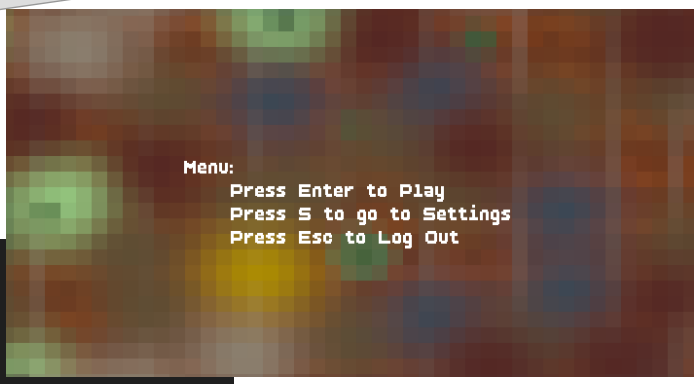




Aktualne statystyki

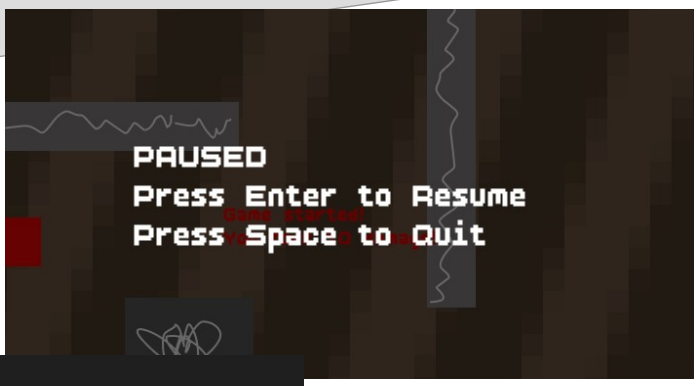
```
// Wyświatlanie info w rogu ekranu:
// fps, czas, hp
leftCornerInfo.setString(
    std::to_string((int)(1/delta)) + " FPS\nTime: "
    + std::to_string((int)timeElapsedSec) + " sec."
    + "\nHP: " + std::to_string(player->getHP())
);
```

Menu




```cpp
MenuScreen::MenuScreen() {
    // Zakładam, że TextureManager działa poprawnie
    try {
        backgroundTexture = TextureManager::get("../assets/menu_bg.png");
        backgroundSprite.setTexture(backgroundTexture);
    }
    catch (...) {
        // Fallback jeśli brak tła
    }

    if (!font.loadFromFile("../assets/font.ttf")) {
        // handle error
    }
    text.setFont(font);
    // Zmieniłem tekst "quit" na "Log Out", żeby pasował do akcji
    text.setString("Menu: \n\tPress Enter to Play\n\tPress S to go to Settings\n\tPress Esc to Log Out");
    text.setCharacterSize(24);
    text.setPosition(350, 250);
}
```

Pauza gry




```
// Tekst ekranu pauzy
pauseText.setFont(font);
pauseText.setString("PAUSED\nPress Enter to Resume\nPress Space to Quit");
pauseText.setCharacterSize(40);
pauseText.setFillColor(sf::Color::White);
// Centrowanie tekstu (dostosowane do dłuższego napisu)
sf::FloatRect textRect = pauseText.getLocalBounds();
pauseText.setOrigin(textRect.left + textRect.width / 2.0f, textRect.top + textRect.height / 2.0f);
pauseText.setPosition(500.f, 300.f);

// Maskownica ekranu w czasie pauzy
pauseOverlay.setSize(sf::Vector2f(1000.f, 600.f));
pauseOverlay.setFillColor(sf::Color(0, 0, 0, 150));
```

Ładowanie poziomu z pliku

```cpp
using json = nlohmann::json;

LevelData LevelLoader::loadFromFile(const std::string& path) {
    std::ifstream file(path);
    if (!file.is_open()) {
        throw std::runtime_error("Nie mozna otworzyc pliku poziomu: " + path);
    }

    json j;
    file >> j;

    LevelData level;
    level.name = j["name"];
    level.size.x = j["size"]["width"];
    level.size.y = j["size"]["height"];
    level.background = j["background"];

    level.playerStart.x = j["player"]["position"][0];
    level.playerStart.y = j["player"]["position"][1];
```

```cpp
// Przeszkody
for (const auto& o : j["obstacles"]) {
    ObstacleData obs;
    obs.bounds = sf::FloatRect(
        o["x"].get<float>(), o["y"].get<float>(),
        o["w"].get<float>(), o["h"].get<float>()
    );
    obs.texture_path = o["texture_path"].get<std::string>();
    level.obstacles.push_back(obs);
}

// Wrogowie
for (const auto& o : j["enemies"]) {
    EnemyData enemy;
    enemy.bounds = sf::FloatRect(
        o["x"].get<float>(), o["y"].get<float>(),
        o["w"].get<float>(), o["h"].get<float>()
    );
    enemy.type = o["type"].get<std::string>();
    level.enemies.push_back(enemy);
}
```

```cpp
48      // Drzwi (NOWE)
49      if (j.contains("doors")) {
50          for (const auto& d : j["doors"]) {
51              DoorData door;
52              door.bounds = sf::FloatRect(
53                  d["x"].get<float>(), d["y"].get<float>(),
54                  d["w"].get<float>(), d["h"].get<float>()
55              );
56              door.next_level_path = d["next_level"].get<std::string>();
57              level.doors.push_back(door);
58          }
59      }
60
61
62      return level;
63  }
```

Mechanika ruchu

```
97  void PlayerBase::applyMovementPhysics(float delta) {
98      // --- 1. FIZYKA RUCHU ---
99
100     // OŚ Y
101     if (inputDirection.y < 0.f) { // W
102         if (speed_vector.y > -max_speed)
103             speed_vector.y -= max_speed * delta;
104     }
105     else if (inputDirection.y > 0.f) { // S
106         if (speed_vector.y < max_speed)
107             speed_vector.y += max_speed * delta;
108     }
109     else if (speed_vector.y > min_speed) {
110         speed_vector.y -= max_speed * delta;
111     }
112     else if (speed_vector.y < -min_speed) {
113         speed_vector.y += max_speed * delta;
114     }
115     else {
116         speed_vector.y = 0.f;
117     }
```

```
119     // OŚ X
120     if (inputDirection.x < 0.f) { // A
121         if (speed_vector.x > -max_speed)
122             speed_vector.x -= max_speed * delta;
123     }
124     else if (inputDirection.x > 0.f) { // D
125         if (speed_vector.x < max_speed)
126             speed_vector.x += max_speed * delta;
127     }
128     else if (speed_vector.x > min_speed) {
129         speed_vector.x -= max_speed * delta;
130     }
131     else if (speed_vector.x < -min_speed) {
132         speed_vector.x += max_speed * delta;
133     }
134     else {
135         speed_vector.x = 0.f;
136     }
```

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