Politechnika Lubelska

Wydział Matematyki i Informatyki Technicznej

Dokumentacja programu

Schizophrenia: The Game

Projekt z zakresu programowania

Autor: Maja Panasiewicz

Lublin 2025

Spis treści

1 Wprowadzenie

Schizophrenia: The Game jest prostą, jednoosobową grą o unikalnym motywie.

1.1 Treść zadania

Zaprojektowanie aplikacji za pomocą środowiska wxWidgets w Code::blocks oraz przygotowanie jej dokumentacji.

1.2 Motto

„Pokonajmy razem schizofrenię!"

1.3 Fabuła gry

Wcielamy się w postać osoby walczącej ze schizofrenią. Napotykamy na swojej drodze różne wytwory naszej choroby, które sprawiają wrażenie miło nastawionych wobec nas, chcą się z nami zaprzyjaźnić. Są to natomiast tylko i wyłącznie pozory; zjawy bowiem wykorzystując swoje zdolności manipulacyjne nakłaniają nas do poczynania niemoralnych rzeczy. Z tego względu chcemy się ich pozbyć. Nie jesteśmy jednak w stanie poradzić sobie w pojedynkę. Potrzebujemy sprzymierzeńców w postaci lekarza, księdza, a także dealera, ponieważ tę walkę możemy wygrać tylko w zgranej ekipie.

1.4 Opis interfejsu

Rysunek 1: Interfejs gry

Zawartość interfejsu:

→ 1. Logo gry

→ 2. Aktualny poziom

→ 3. Wskazówka

→ 4. Wygląd zjawy

→ 5. Statystyki zjawy

→ 6. Statystyki naszej postaci

→ 7. Klasa naszej postaci, imię

→ 8. Przyciski do kupna następnej postaci

→ 9. Aktualny stan pieniędzy

2 Kod źródłowy

2.1 Klasy

```
class Level
{
private:
  int currentLevel;
  static const int maxLevel = 5;
public:
  Level() : currentLevel(1) {}

  void nextLevel()
  { if (currentLevel < maxLevel)
    {currentLevel++;} }
  int getLevel() const
  {return currentLevel;}
};
```

Kod 1: Level

Przedstawia on klasę poziomu, oraz jego 2 metody: nextLevel, getLevel które służą do podania aktualnego poziomu gracza oraz przeskoczenia na następny po wygranej turze.

```
class Postac
{
public:
  int hp;
  int dmg;
  int armor;

  Postac(int h, int d, int a) : hp(h), dmg(d), armor(a) {}
  virtual ~Postac() {}
  virtual const char* typPostaci() const = 0;

  virtual void dodajHP(int amount)
  {hp += amount;}

  virtual void dodajDMG(int amount)
  {dmg += amount;}

  virtual void dodajARMOR(int amount)
  {armor += amount;}

  bool walka(Boss& boss, const std::vector<Postac*>& postacie);
};
```

Kod 2: Postać

Klasa postaci zawiera jej statystyki: punkty życia (ang. health points), obrażenia od ataku (ang. damage) oraz pancerz (ang. armor). Mamy też typ postaci oraz łatwe metody ulepszające ich statystyki. Metoda walka jest opisana później przy sekcji metod.

```
class Boss : public Postac
{
public:
  Boss(int h=0, int d=0, int a=0) : Postac(h, d, a) {}
  const char* typPostaci() const override
  {return "Boss";}

  static Boss* stworzBossa(int randomIndex, int level);
};

class Ksiadz : public Postac
{
  public:
  Ksiadz(int h, int d, int a) : Postac(static_cast<int>(h * 1.1), d, static_cast<int>(
      a * 1.7)) {}

  const char* typPostaci() const override
  {return "Ksiadz";}

  void dodajHP(int amount) override
  {hp += amount;}

  void dodajDMG(int amount) override
  {dmg += amount;}

  void dodajARMOR(int amount) override
  {armor += static_cast<int>(amount * 2);}
};
```

Kod 3: Boss oraz ksiądz

Tu są przykładowo klasy pochodne po klasie postać, czyli Boss oraz ksiądz.
Boss posiada dodatkowo metodę stworzBossa ukazaną później.
Ksiądz jak widzimy ma zmienione przeliczniki przy kupnie pancerza, ale też bazowe statystyki, ze względu na to, że wylosowany na starcie gry nie dawał sobie rady.
Klasami pochodnymi są również dealer, lekarz oraz NN (klasa dla postaci jeszcze nieistniejącej).

```
class Gold
{
public:
  int gold;
Gold(int g) : gold(g){};

  int getGold() const
  {return gold;}

  void setGold(int g)
    {gold = g;}
};
```

Kod 4: Gold

Klasa gold jest naszym portfelem, bazują na niej wszystkie operacje kupna w grze. Zawiera proste metody getGold - zwracającą aktualny stan pieniędzy, oraz setGold - ustawiającą nam nowy stan.

2.2 Metody

```
bool Postac::walka(Boss& boss, const std::vector<Postac*>& postacie) {
  int sumaHP = 0;
  int sumaDMG = 0;
  int sumaARMOR = 0;

  for (const auto& postac : postacie) {
    if (postac) { // Sprawdza czy wskaznik nie jest nullptr
      sumaHP += postac->hp;
      sumaDMG += postac->dmg;
      sumaARMOR += postac->armor;}
    }

  int wynikPostaci = sumaHP - (boss.dmg * (100 - sumaARMOR) / 100);
  int wynikBossa = boss.hp - (sumaDMG * (100 - boss.armor) / 100);

  return wynikPostaci <= wynikBossa; // Zwraca 1, jesli boss wygra
}
```

Kod 5: Walka z bossem

Tutaj metoda walki z bossem. Pobiera ona nasz wektor z postaciami, aby zsumować ich statystyki do finalnego wyniku starcia. Logika stojąca za tym bardzo prosta: warto zwrócić uwagę na mocne i słabe strony stworzenia z którym będziemy walczyć, oraz postaci którymi chcemy go pokonać.

```
void PokazObrazek(const wxString& sciezkaDoObrazka, const wxString& sciezkaDoDzwieku)
    {

  wxFrame* oknoObrazka = new wxFrame(nullptr, wxID_ANY, "", wxDefaultPosition,
      wxDefaultSize,
  wxFRAME_NO_TASKBAR | wxSTAY_ON_TOP | wxFRAME_SHAPED);

  wxImage obrazek;
  if (!obrazek.LoadFile(sciezkaDoObrazka)) {
    wxLogError("Nie udalo sie zaladowac obrazu z podanej sciezki.");
    return;
  }
  // Tworzenie bitmapy z zaladowanego obrazu
  wxBitmap bitmapa(obrazek);

  // Pobierz wymiary ekranu
  wxDisplay display;
  wxRect ekran = display.GetGeometry();
  int ekranWidth = ekran.GetWidth();
  int ekranHeight = ekran.GetHeight();

  // Ustaw rozmiar okna na pelny ekran
  oknoObrazka->SetClientSize(ekranWidth, ekranHeight);
  oknoObrazka->Move(0, 0);

  PlaySound(sciezkaDoDzwieku.c_str(), NULL, SND_FILENAME | SND_ASYNC);

  // Tworzenie kontrolki z obrazkiem
  wxStaticBitmap* kontrolkaObrazka = new wxStaticBitmap(oknoObrazka, wxID_ANY, bitmapa
      );

  // Zablokowanie zamykania okna
  oknoObrazka->Bind(wxEVT_CLOSE_WINDOW, [](wxCloseEvent& event) {
    event.Veto();
  });

  oknoObrazka->Show(true); }
```

Kod 6: Wyskakujący obrazek

```
int losuj() {
  return rand() % 3;
}

Postac* stworzRandomPostac() {
  srand(time(0));
  int typPostaci = losuj();
  switch (typPostaci) {
    case 0: return new Ksiadz(100, 60, 10);
    case 1: return new Diler(100, 60, 10);
    case 2: return new Lekarz(100, 60, 10);
  }
}

Postac* stworzPostac(int typPostaci) {
  switch (typPostaci) {
    case 0: return new Ksiadz(100, 60, 10);
    case 1: return new Diler(100, 60, 10);
    case 2: return new Lekarz(100, 60, 10);
    case 3: return new NN(0,0,0);
  }
}
```

Kod 7: Losowanie, tworzenie postaci

Tworzenie losowej postaci jest nam potrzebne na samym początku gry, bowiem gracz musi wiedzieć, że schizofrenia nie jest łatwa do pokonania. Poźniej na szczęście wygląd drużyny jest już tylko w naszych rękach. W tym celu tworzymy postać o klasie którą sobie wybierzemy przy zakupie postaci. Na pozór nowe postacie tworzą się z takimi samymi statystykami, jednak trzeba pamiętać, że ich klasy mają bazowe przeliczniki dla wybranych statystyk, co sprawia, że sytuacyjnie będą słabsze lub silniejsze.

```
Boss* Boss::stworzBossa(int randomIndex, int level) {
  int hp, dmg, armor;
  int multiplier = level;
  switch(randomIndex) {
    case 0:
    hp = 130 * multiplier; dmg = 50 * multiplier; armor = 30 * multiplier;
    break;
    case 1:
    hp = 150 * multiplier; dmg = 40 * multiplier; armor = 15 * multiplier;
    break;
    case 2:
    hp = 100 * multiplier; dmg = 105 * multiplier; armor = 20 * multiplier;
    break;
  }
  return new Boss(hp, dmg, armor); }
```

Kod 8: Tworzenie bossa

Jak widać, nasza zmora również jest losowana, ale to nie wszystko. Żeby nie było nudy, każda z nich jest inna, a z każdym poziomem stają się coraz gorsze do uporania. Przy opracowaniu odpowiedniej strategii oraz sprzyjającym szczęściu grę na szczęście da się wygrać. Niestety ma się to nijak do rzeczywistości, gdzie ta choroba nie jest w żadnym scenariuszu uleczalna.

```
void ustawImieKsiadz(wxTextCtrl* kontrolka){
  // Lista dostepnych imion
  std::vector<std::string> imiona = {"Izajasz", "Jakub", "Piotr","Dawid", "Jan", "
      Szymon","Filip","Tomasz"};

  // Inicjalizacja generatora losowego
  static bool initialized = false;
  if (!initialized) {
    srand(time(nullptr));
    initialized = true;
  }

  // Losowanie indeksu
  int losowyIndeks = rand() % imiona.size();

  // Ustawienie losowego imienia w podanej kontrolce
  kontrolka->SetLabel(imiona[losowyIndeks]);
}
```

Kod 9: Ustawianie losowych imion postaci

Tutaj myślę, że w komentarzach przy kodzie wszystko jest zgrabnie wyjaśnione. W przypadku gdyby postaciom wylosowane imiona się powtarzały nie ma co się martwić - możemy je ręcznie zmienić, bo znajduje się tam na pierwszy rzut oka niewidoczny wxTextCtrl.

2.3 Główny kod programu

```
Level stgLevel;
Postac* postac1 = stworzRandomPostac();
Postac* postac2 =stworzPostac(3);
Postac* postac3 =stworzPostac(3); //przykladowo tworzymy NN
Postac* postac4 =stworzPostac(3);
Postac* postac5 =stworzPostac(3);
Boss* bossWSK = bossMap[stgLevel.getLevel()]; //tworzy boss1 na start
Gold g1(3);
```

Kod 10: Inicjalizacja podstawowych zmiennych

Inicjalizacja zmiennych na samym początku kodu pozwala mi na swobodne korzystanie z nich gdziekolwiek indziej.
Tworzę poziom (ang. level), pięć postaci typu "NN", bossa dla pierwszego poziomu gry, oraz 3 pieniążki na start.

```
void stgDialog::OnInit(wxInitDialogEvent& event)
{
  if (bossWSK)
  {delete bossWSK;}

  if (postac1)
  {
    if (dynamic_cast<Ksiadz*>(postac1))
    {p1bitmap->SetBitmap(czempy[0]);
      ustawImieKsiadz(CzempImie);}

    else if (dynamic_cast<Diler*>(postac1))
    {p1bitmap->SetBitmap(czempy[1]);
      ustawImieDiler(CzempImie);}

    else if (dynamic_cast<Lekarz*>(postac1))
    {p1bitmap->SetBitmap(czempy[2]);
      ustawImieLekarz(CzempImie);}}

  int randomIndex = rand() % 3;
  wxMessageBox(slowa[randomIndex], _("Poznajmy sie!"), wxOK | wxICON_INFORMATION);
  bossWSK = Boss::stworzBossa(randomIndex,stgLevel.getLevel());
  bossMap[stgLevel.getLevel()] = bossWSK;
  BossBitmap->SetBitmap(bossy[randomIndex]);

  LevelPNG->SetBitmap(leveleBit[0]);

  wxString a=wxString::Format("%d",bossWSK->hp);
BossHP->SetLabel(a);
  wxString b=wxString::Format("%d",bossWSK->dmg);
BossDMG->SetLabel(b);
  wxString c=wxString::Format("%d",bossWSK->armor);
BossARM->SetLabel(c);
  wxString x1=wxString::Format("%d",postac1->hp);
HP->SetLabel(x1);
  wxString y1=wxString::Format("%d",postac1->dmg);
DMG->SetLabel(y1);
  wxString z1=wxString::Format("%d",postac1->armor);
ARMOR->SetLabel(z1);
  wxString q=wxString::Format("%d",g1.getGold());
  ileGolda->SetLabel(q);

  p2bitmap->SetBitmap(czempy[3]);
  p3bitmap->SetBitmap(czempy[3]);
  p4bitmap->SetBitmap(czempy[3]);
  p5bitmap->SetBitmap(czempy[3]);
}
```

Kod 11: Start programu - inicjalizacje

Najpierw usuwam wskaźnik do bossa którego utworzyłam na starcie, ponieważ chcę stworzyć nowego - wylosowanego, oraz ustawić mu odpowiedni wygląd i przywitanie. Ustawiam też wygląd oraz imię dla losowej postaci którą zaczniemy tą grę. Ta część kodu ma głównie na celu prawidłowe ustawienie obrazków oraz wyświetlanych tekstów.

```
void stgDialog::OnBossBitmapClick(wxCommandEvent& event)
{
  std::vector<Postac*> druzyna = {postac1, postac2, postac3, postac4, postac5};
  // Usun puste wskazniki (jesli np. gracz kupil tylko 2 postacie)
  druzyna.erase(std::remove(druzyna.begin(), druzyna.end(), nullptr), druzyna.end());

  Boss* bossWSK = bossMap[stgLevel.getLevel()];

  if(stgLevel.getLevel()==5 && !druzyna.empty() && postac1->walka(*bossWSK, druzyna)
     == 0)
  {PokazObrazek("pics\\yay2.png","pics\\add_done.wav");}
  else if (!druzyna.empty() && postac1->walka(*bossWSK, druzyna) == 0)
  {
    g1.setGold(g1.getGold() + 5);
    stgLevel.nextLevel();

    int K=(stgLevel.getLevel()-1);
    LevelPNG->SetBitmap(leveleBit[K]);

    int randomIndex = rand() % 3;
    int randomInd = ((randomIndex)+3*(stgLevel.getLevel()-1));
    bossWSK = Boss::stworzBossa(randomIndex,stgLevel.getLevel());
    bossMap[stgLevel.getLevel()] = bossWSK;
    BossBitmap->SetBitmap(bossy[randomInd]);
    wxMessageBox(slowa[randomInd], _("Poznajmy sie!"), wxOK | wxICON_INFORMATION);

    wxString a=wxString::Format("%d",bossWSK->hp);
  BossHP->SetLabel(a);
    wxString b=wxString::Format("%d",bossWSK->dmg);
  BossDMG->SetLabel(b);
    wxString c=wxString::Format("%d",bossWSK->armor);
  BossARM->SetLabel(c);
    wxString q=wxString::Format("%d",g1.getGold());
  ileGolda->SetLabel(q);}
else if (postac1->walka(*bossWSK, druzyna) == 1)
  {PokazObrazek("pics\\jeff2.png","pics\\fnafsound.wav");}
}
```

Kod 12: Walka, po kliknięciu na obrazek zjawy

Tutaj mamy 3 możliwe rozwiązania sytuacji. Możemy walkę ze schizofrenią przegrać, wtedy wyskakuje nam na cały ekran Jeff The Killer [1] oraz krzyki z gry Five Nights at Freddy's [2].Możemy też walkę wygrać, a jeśli uda nam się to zrobić na ostatnim poziomie, to skończymy grę i zobaczymy uśmiechnięte postacie z gry matematycznej dla dzieci „Wesołe Miasteczko", oraz usłyszymy oklaski z gry „Stare Zamczysko", wydanych wiele lat temu przez YoungDigitalPlanet.
Ostatnią opcją jest wygranie walki na poziomach 1-4. Wtedy dostajemy 5 monet na ulepszanie i kupowanie postaci, oraz nowego przeciwnika adekwatnego do naszego poziomu. Wszystkie kontrolki ustawiamy tak, aby odpowiadały rzeczywistemu stanu rzeczy.

[1] link do creepypasty: `pasta.fandom.com/pl/wiki/Jeff_The_Killer`
[2] poczytaj więcej: `freddy-fazbears-pizza.fandom.com`

```
void stgDialog::OnaddHPClick(wxCommandEvent& event)
{
  if (g1.getGold() <= 0)
  {wxMessageBox(_("Operacja niedozwolona! Za malo zlota."), "noob", wxOK |
      wxICON_ERROR);
    return;}
  postac1->dodajHP(10);
  g1.setGold(g1.getGold() - 1);

  wxString x1 = wxString::Format("%d", postac1->hp);
  HP->SetLabel(x1);
  wxString q=wxString::Format("%d",g1.getGold());
  ileGolda->SetLabel(q);
}
```

Kod 13: Ulepszanie statystyk

Ten kod jest dość prosty: Jeśli nie mamy wystarczająco złota, nie możemy wykonać akcji. Jeśli jednak stać nas na ulepszenie, dodajemy 10 punktów życia pierwszej postaci, po czym zabieramy sztukę złota. W zależności od klasy postaci dostaniemy bazową wartość lub większą, przykładowo:

Kupując dla lekarza 10 punktów życia, otrzyma on 12.

Kod wygląda analogicznie dla obrażeń od ataku (+7) oraz pancerza (+8).

```
void stgDialog::OnbuyKsiadzClick(wxCommandEvent& event)
{
if (g1.getGold() < 3)
{wxMessageBox(_("Operacja niedozwolona! Za malo zlota."), "noob", wxOK | wxICON_ERROR)
    ;
  return;}
if(stgLevel.getLevel()==1)
{
  wxMessageBox(_("Nie mozesz jeszcze kupowac postaci!"), (_("zlamas")), wxOK |
      wxICON_ERROR);
  return;
}

if (dynamic_cast<NN*>(postac2))
{
  delete postac2; // Usuwamy stara, zla postac
  postac2 = nullptr;

  postac2 = stworzPostac(0);

  postac2->hp = static_cast<int>(postac2->hp * 1.1);
  postac2->dmg = static_cast<int>(postac2->dmg * 1.1);
  postac2->armor = static_cast<int>(postac2->armor * 1.1);

  HP2->SetForegroundColour(*wxWHITE);
  DMG2->SetForegroundColour(*wxWHITE);
  ARMOR2->SetForegroundColour(*wxWHITE);
  CzempImie2->SetForegroundColour(*wxWHITE);
  addHP2->SetBitmap(guziki[0]);
  addDMG2->SetBitmap(guziki[1]);
  addARMOR2->SetBitmap(guziki[2]);

  ustawImieKsiadz(CzempImie2);
  p2bitmap->SetBitmap(czempy[0]);

  wxString x2 = wxString::Format("%d", postac2->hp);
HP2->SetLabel(x2);
  wxString y2 = wxString::Format("%d", postac2->dmg);
DMG2->SetLabel(y2);
  wxString z2 = wxString::Format("%d", postac2->armor);
ARMOR2->SetLabel(z2);
}
else if (dynamic_cast<NN*>(postac3))
  {delete postac3;
  postac3 = nullptr;
  ... //wszystko idzie tak samo jak dla postaci2 az do postaci5
  }
else
  {wxMessageBox(_("Masz juz maksymalna liczbe postaci!"), "Info", wxOK |
      wxICON_INFORMATION);
  return;}

// Odejmujemy zloto
  g1.setGold(g1.getGold() - 3);
  wxString q = wxString::Format("%d", g1.getGold());
  ileGolda->SetLabel(q);}
```

Kod 14: Kupno nowej postaci

Tutaj, jeżeli nie mamy drugiego poziomu albo wystarczająco pieniędzy nie możemy kupić nowej postaci. W przeciwnym przypadku kasujemy postac2 typu „NN”i tworzymy nową - tutaj księdza. Będzie jednak trochę silniejsza niż bazowy ksiądz, a każda następna postać będzie miała przelicznik na statystykach coraz to większy. Potem ustawiamy widoczność wszystkich kontrolek oraz to co przedstawiają. Gdy mamy już 5 postaci nie możemy kupić ich więcej. Ustawiamy pieniądze do aktualnego stanu.

3 Podział pracy

Będąc szczera, sporą część kodu pomógł mi pisać ChatGPT.
Całą resztę pracy, tzn: Interfejs GUI, grafikę użytą w grze, przygotowanie dokumentacji, a przede wszystkim pomysł zawdzięczam tylko i wyłącznie sobie.

4 Przemyślenia

Myślę, że sama gra w sobie ma jeszcze bardzo duże pole do popisu i może być traktowana jako jedynie „podstawa"do stworzenia czegoś większego. Nie mówię tutaj o wyglądzie - który był w zamierzeniu tandetny i przypominający słabe, głupkowate gry gatunku straszaków. Mogłaby być wprowadzona chociażby mechanika walki przypominająca wymianę ciosów. Dodanie animacji zadania obrażenia zmorze też byłaby dobrze wyglądającym rozwiązaniem. Mapa jako globalne wprowadzenie małych zmian (np. w parzystych rundach dostaniemy więcej pieniędzy) wielokrotnie przechodziła przez myśl. Następne poziomy, lepsze cytaty czy bonusy przy wielokrotnym ulepszeniu tej samej postaci są swego rodzaju oczywistościami. Wtedy jednak ten projekt zająłby mi nieskończenie wiele czasu, ponieważ mogłabym wymyślać nowe opcje aż do zakucia w grobowe deski. Mimo wszystko jestem ze swojej pracy bardzo zadowolona, głównie ze względu na ilość wiedzy którą w międzyczasie przyswoiłam.

5 Podziękowania

Pomimo tego, że projekt przygotowywałam indywidualnie, jest parę osób którym wiszę wspomnienie.

Szczególnie chciałabym w tym miejscu podziękować mojej przyjaciółce z pokoju, Ani Wszole. Jako osoba doświadczająca różnych ciekawych zjawisk (pół żartem, pół serio) była ona moją chodzącą inspiracją dla tego projektu.

Dla mojej największej motywacji, kochanego i wspierającego przy upadkach Macieja Michalaka.

Dla dr Pawła Szymona Wlazia, za smaczne czekoladki Merci (po raz pierwszy miałam okazji spróbować smaków z kawowej bombonierki).