

## Przykładowe tematy projektowe

### Urząd pocztowy

Zasymuluj działanie urzędu pocztowego, w którym funkcjonuje  $n$  okienek obsługujących różne (niekoniecznie rozłączne) zbiory zleceń. Przyjmij, że klienci oczekują na obsługę w  $m$  kolejkach,  $n/2 < m < n$ . Każda z kolejek oczekuje pod wybranym okienkiem lub „rozchodzi się” do dwóch wybranych okienek. Jeśli klient ma kilka zleceń do realizacji, to nie można wykluczyć, że będzie musiał oczekiwać w więcej niż jednej kolejce.

### Poszukiwanie miejsca zerowego funkcji

Zilustruj graficznie poszukiwanie miejsca zerowego funkcji dwiema różnymi metodami. Funkcja może być dowolną funkcją złożoną, konstruowaną z szerokiego repertuaru funkcji bazowych. Miejsce zerowe wskaż z zadaną dokładnością  $\epsilon$ . Zbadaj eksperymentalnie zbieżność obu metod.

### Poczta kurierska

Porównaj kilka (co najmniej trzy) metody wyszukiwania optymalnej trasy przejazdu dla kuriera:

- rozwożącego paczki przechowywane w magazynie A pod wskazane adresy  $A_1, A_2, \dots, A_n$ ,
- powracającego po rozwiezieniu paczek do punktu wyjścia A.

Wśród metod ma się znaleźć metoda oparta na algorytmie genetycznym.

Przy porównaniu metod weź pod uwagę następujące kryteria:

- skuteczność metody w wyszukiwaniu rozwiązań optymalnych,
- złożoność czasową metody,
- złożoność pamięciową metody.

### Szeregowanie zadań na procesorach

Porównaj pod względem złożoności czasowej i pamięciowej trzy algorytmy szeregowania zadań na procesorach, w tym RR i algorytm wielokolejkowy ze zmiennymi priorytetami.

Wylicz narzuty czasowe związane z realizacją samego procesu szeregowania zadań. Sprawdź zależność tych narzutów od specyfiki zadań. Spróbuj zaprojektować ciekawy interfejs graficzny dla programu.

### Algorytm genetyczny

Zaprojektuj algorytm genetyczny znajdowania miejsc zerowych zadanej funkcji jednej zmiennej rzeczywistej. Funkcja może być dowolną funkcją złożoną, konstruowaną z

szerokiego repertuaru funkcji bazowych. Zastosuj zróżnicowane metody selekcji. Poszukaj tych miejsc we wskazanym przedziale  $\langle a ; b \rangle$ .

### **Gra w statki**

Zaprojektuj i zaimplementuj aplikację przeznaczoną do gry w statki. Zaplanuj dwa alternatywne tryby gry:

- tryb Hot Seat dla dwóch graczy,
- tryb „gracz przeciwko komputerowi”, na różnych poziomach trudności.

Aplikacja winna być zrealizowana z wykorzystaniem elementów sztucznej inteligencji (m.in. przewidywanie ustawienia statków przeciwnika i ruchów przeciwnika na podstawie dotychczasowego przebiegu rozgrywki).

Aplikacja winna posiadać interfejs graficzny, oparty na wykorzystaniu biblioteki SFML.

### **Labirynt**

Zbuduj aplikację do generacji labiryntów o różnych poziomach trudności.

Zaproponuj różne tryby wyświetlania zaprojektowanego labiryntu (interfejs graficzny!).

Rozbuduj aplikację o funkcję/funkcje „przechodzenia labiryntu”.

### **Gra "Warcaby"**

Zrealizuj aplikację do gry w warcaby.

Obowiązkowa funkcjonalność aplikacji:

- gra dostępna w dwóch trybach: rozgrywki turowej pomiędzy dwoma graczami i rozgrywki z przeciwnikiem komputerowym,
- możliwość wglądu w statystyki gry: bilans wygranych i przegranych potyczek, najlepszy czas pojedynku, liczba zбитych pionów, największą liczbą zбитych pionów w jednej serii.

Ogólne zasady gry: rozmiar planszy 8x8; dama może przesunąć się o dowolną liczbę pól w jednym ruchu; damę może zbić zwykły pionek; gracz ma obowiązek wykonania (dowolnego) bicia; bicie zwykłym pionem do tyłu jest dozwolone; podczas rozgrywki turowej obowiązuje limit czasowy ruchu; po przekroczeniu limitu czasu będzie wykonywany ruch losowy.

Wyposaź aplikację w odpowiedni interfejs graficzny.

### **Problem komiwojażera**

Rozwiąż problem komiwojażera przy wykorzystaniu algorytmu genetycznego. Zaprojektuj odpowiednie funkcje selekcji, krzyżowania i mutacji. Zastosuj właściwe środki graficzne do wizualizacji procesu odnajdowania najkrótszej drogi w wylosowanym grafie.

Opcjonalna funkcjonalność aplikacji:

- porównaj efektywność i wydajność przedmiotowego algorytmu z efektywnością i wydajnością algorytmu „brute force”.

### **Gra w ping-ponga**

Napisz oryginalną aplikację przeznaczoną do gry w ping-ponga.

Obowiązkowa funkcjonalność aplikacji:

- dwa tryby rozgrywki:
  1. Gracz vs Gracz
  2. Gracz vs Komputer (projekt własnego modułu SI),
- implementacja praw fizyki, m.in. grawitacji, oporu powietrza, rozkładu sił w zależności od mocy i kierunku uderzenia,
- grafika 2D,
- rozbudowane menu, uwzględniające możliwość zmiany parametrów fizycznych rozgrywki.

### **Gra RougeLike**

Zaprojektuj i zaimplementuj oryginalną 2-wymiarową grę losową z gatunku rouge-like (zręcznościową), w której gracz wciela się jedną z kilku dostępnych postaci.

Cel gry:

Gracz ma za zadanie przechodzić kolejne, rozmieszczone na planszy pokoje, które prowadzą do nowych, coraz trudniejszych poziomów. Przejście na kolejny poziom wymaga pokonania jednego z głównych przeciwników, znajdującego się w losowym, odpowiednio oznaczonym pokoju. Gracz przemierza trasę w poszukiwaniu tego pokoju, "czyszcząc" obowiązkowo wszystkie pokoje, które znajdują się na tej trasie (wyczyszczenie pokoju polega, w zależności od jego rodzaju, na pokonaniu wszystkich znajdujących się tam przeciwników lub na rozwiązaniu prostej łamigłówki). Po drodze gracz ma możliwość odnalezienia przedmiotów, które zwiększą jego statystyki.

Celem gry jest dotarcie na najniższy możliwy poziom. Rozgrywka kończy się po upływie zadanego czasu.

Obowiązkowa funkcjonalność aplikacji:

- sterowanie postacią za pomocą klawiatury,
- obliczanie i zadawanie obrażeń na podstawie statystyk podstawowych i statystyk przedmiotów,
- dostępność kilku postaci oraz przeciwników o różnych mechanikach (np. wrogowie skaczący na gracza, wrogowie strzelający, itp.),
- losowa generacja poziomów z różnymi pokojami (w każdym pokoju znajduje się co najmniej jedno przejście oraz jeden przedmiot; na całym poziomie znajduje się dokładnie jeden przeciwnik główny),
- dodatkowy tryb gry, polegający na pokonywaniu jak największej ilości przeciwników pojawiających się w falach (plansza bez pomieszczeń, losowy przedmiot znajduje się na końcu fali),
- zróżnicowana mechanika ataków, zależna od typu postaci (ataki zasięgowe i broń biała).

Opcjonalna funkcjonalność aplikacji:

- kooperacja dwóch graczy na zasadzie wyboru dwóch postaci i równoczesnego sterowania obiema postaciami za pomocą tej samej klawiatury (w takim wypadku przeciwnicy będą odpowiednio silniejsi),
- przedmioty użytkowe, czyli przedmioty niezwiększające statystyk; użycie tych przedmiotów wiąże się z pewnymi korzyściami, np. użycie klepsydry może zagwarantować chwilowe spowolnienie czasu.

### **AUTO – program zarządzający flotą autonomicznych samochodów, jeżdżących na terenie Poznania, połączony z symulatorem pojazdów**

Główne założenia:

- utworzenie programu przyjmującego od klientów zlecenia na kursy autonomicznych samochodów i przydzielającego je do konkretnych pojazdów oraz monitorującego status tych zleceń; aplikacja będzie działać na serwerze z systemem Debian (technologie: C++, cppcms, cURLpp, Google Maps API, nginx),
- opracowanie i zaimplementowanie algorytmu przydzielania tras w taki sposób, żeby zmaksymalizować ilość wykonanych kursów przy danej flocie pojazdów,
- utworzenie symulatora pojazdów, który będzie się komunikował z głównym systemem i realnie odwzorowywał zachowanie floty (technologie: C++, cURLpp, Google Maps API),
- utworzenie symulatora zleceń od klientów, który za pomocą żądań HTTP będzie je wysyłał do głównego systemu (technologie: C++, cURLpp, Google Maps API),
- utworzenie niewielkiej aplikacji przeglądarkowej, pozwalającej na podgląd mapy z zaznaczonymi pojazdami, ich statusu oraz wprowadzania zleceń (technologie: TypeScript, Google Maps JS API),
- implementowanie wszystkich elementów pozwalające na przyszłą rozbudowę projektu.

### **Gra typu arkanoid**

Założenia gry:

- przesuwająca się paletka, kontrolowana myszką, uderza w piłeczkę, która odbija się od ścian lub umieszczonych na planszy cegiełek;
- cegiełki charakteryzowane są właściwościami, takimi jak: ilość uderzeń potrzebna do zbitcia cegiełki, rozmiar cegiełki, zawarte bonusy; w wyniku uderzenia piłeczki w cegiełkę, losowo wybrane lub wcześniej ustalone w edytorze bonusy (w postaci dodatkowej piłki, większej paletki, zmiany prędkości piłki) wydobywają się z niej i biorą udział w dalszej grze;
- piłeczka porusza się zgodnie z zasadą fizyki dotyczącą kąta padania - kąta odbicia.

Cel gry:

- zbitcie wszystkich cegiełek znajdujących się na planszy,
- niedopuszczenie do spadku piłki poniżej paletki.

Planowana funkcjonalność aplikacji:

- edytor poziomów gry, umożliwiający ustawianie na planszy cegiełek (za pomocą myszki) i określenie ich właściwości,
- efekty dźwiękowe i muzyka w tle,
- interfejs graficzny.