Liszaj

Dokumentacja

Projekt z Programowania w języku C++

Brajan Miśkowicz, P4, 163976

Opis założeń:

Problem liszaja to gra symujacyjna przedstawiająca tablicę n x n komórek skóry w której środkowa zostaje zarażona. Po upływie kolejnej jednostki czasu zarażona komórka ma 50% szans zarażenia każdej ze zdrowych sąsiednich komórek. Po sześciu jednostkach czasu zarażona komórka staje się na cztery jednostki odporna i następnie zdrowa. Następuje symulacja choroby po kolejnych jednostkach czasu, pokazująca zawartość tablicy.

Ogólny zarys logiki działania symulacji:

Symulacja rozpoczyna się z przykładowymi wymiarami 11 x 11 komórek z zarażoną środkową komórką, istnieją więc dwie tablice jednowymiarowe o wielkości 121 – jedna mieszcząca stany komórek, a druga pozostały czas aktualnych stanów komórek.
Stany komórek oznaczane są następująco:
0 – komórka zdrowa
1 – komórka chora
2 – komórka odporna
Tablice wyglądają na początku więc w taki sposób:

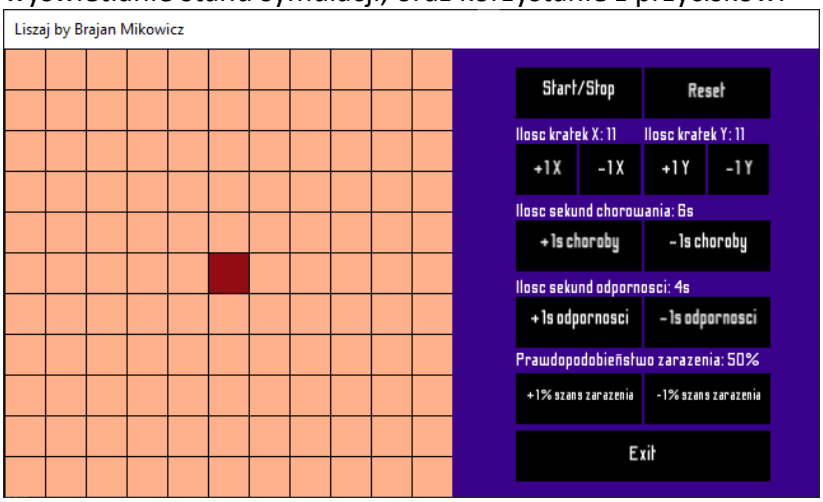
Tablica stanów:

```
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
```

Tablica czasów:

```
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 6 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
```

Najważniejszy algorytm symulacji zajmuje się zmianami stanów tych dwóch tablic – każda chora komórka próbuje zarazić sąsiednie zdrowe komórki, a algorytm sprawdza czy minął czas trwania stanu każdej kolejnejkomórki i tak jest to zmniejsza go o jeden co jednostkę czasu, jeżeli skończył się czas trwania stanu i komórka jest chora, lub odporna, to następuje zmiana stanu na kolejny.
Sam projekt zrealizowany jest z pomocą biblioteki SFML, co pozwala na dużo czytelniejsze wyświetlanie stanu symulacji, oraz korzystanie z przycisków:



Opis działania i obsługi grafiki opisany jest szczegółowo w dołączonym pliku Instrukcja.pdf

Klasy:

Liszaj – najważniejsza klasa projektu, zawiera główną metodę, zajmuje się oknem aplikacji i interakcją z użytkownikiem. Posiada następujące metody:

- prywatne:
• *void Okno()* – wczytuje grafiki komórek, inicjalizuje okno i początkowe właściwości kratek.
• *void Reset()* – resetuje ona symulację, tj. środkowa komórka staje się świeżo zarażona, a pozostałe zdrowe, wywoływana głównie po kliknięciu odpowiednich przycisków.
• *void Change()* – zmienia ilość komórek na wysokość i szerokość zależnie od zmiennych *heigth* i *width*, wywoływana po kliknięciu odpowiednich przycisków.

- publiczne:
• *Liszaj()* – konstruktor klasy, tworzy tablice czasów, stanów i zapasową stanów, pobiera z pliku czcionkę do napisów, tworzy wszystkie przyciski i napisy, wywołuje funkcję *Okno.*
• ~Liszaj*()* – destruktor klasy, usuwa tablice, przyciski, napisy i okno.
• *void Start()* – posiada główną pętlę symulacji, działając na zegarze z biblioteki sfml co pół sekundy wywołuje zmianę stanów komórek, zmiane ich grafik i ich wyrysowanie.
• *void Render()* – metoda zajmująca się przedstawianiem elementów graficznych biblioteki SFML, wywoływana po wszelakich zmianach czyści okno i tworzy wszystkie elementy graficzne ponownie.
• *void Grafika()* – obsługuje wszystkie zdarzenia dotyczące okna: pozwala na obsługę przycisków i wywołuje zmiany nimi spowodowane, kontroluje zmianę wielkości okna i wielkości elementów wewnątrz okna podczas takich zmian, pozwala na tzw. interakcję z komórkami, czyli zmianę stanu komórki po kliknięciu w nią myszą.

Ponadto klasa Liszaj posiada następujące zmienne wykorzystywane w metodach:

- prywatne:
• *RenderWindow *window* – wskaźnik do obiektu klasy RenderWindow z biblioteki graficznej SFML, jest to główne okno programu, rysowane na nim są wszystkie elementy.
• *Event event* – obiekt klasy Event z biblioteki SFML, jest to zdarzenie związane z oknem programu.
• *Buttons* exit* – wskaźnik do obiektu opisanej niżej klasy *Buttons*, przycisk kończący działanie programu.
• *Buttons* start* – przycisk rozpoczynający/zatrzymujący symulację.
• *Buttons* reset* – przycisk resetujący stany symulacji (zarażona tylko środkowa komórka).
• *Buttons* sick, Buttons* sick2* – przyciski zwiększający i zmniejszający czas chorowania komórek.
• *Buttons* immunity, Buttons* immunity2* – przyciski zwiększający I zmniejszający czas odporności komórek po chorobie.
• *Buttons* probability, Buttons* probability2* - przyciski zwiększający i zmniejszający prawdopodobieństwo zarażenia sąsiedniej komórki przez chorą.
• *Buttons* numberX, Buttons* numberX2* - przyciski zwiększający I zmniejszający ilość komórek na szerokość.

• *Buttons* numberY, Buttons* numberY2* - przyciski zwiększający I zmniejszający ilość komórek na wysokość.
• *Strings* s_probability* – wskaźnik do obiektu opisanej niżej klasy *Strings*, napis mówiący, jakie aktualnie jest prawdopodobieństwo zarażenia sąsiedniej komórki przez chorą.
• *Strings* s_immunity* – napis mówiący jaki jest aktualnie czas odporności po chorobie komórek.
• *Strings* s_sick* – napis mówiący jaki jest aktualnie czas choroby komórek.
• *Strings* s_size* – napis mówiący ile jest aktualnie komórek na wysokość i szerokość.
• *Font font* – obiekt klasy Font, przechowuje czcionkę pobraną z pliku potrzebną do wszystkich napisów.
• *bool stop* – zmienna mówiąca o tym, czy symulacja jest zatrzymana (ma wówczas wartość 1), czy działająca (wartość 0).

- publiczne:
• *Tilemap map* – obiekt klasy Tilemap opisanej niżej, przechowuje tekstury komórek.
• *int* states* – tablica przechowująca stany komórek.
• *int* statescopy* – tablica pomocnicza przechowująca stany komórek z poprzedniej tury.
• *int* times* – tablica przechowująca pozostałe pozostałe czasy stanów danych komórek.

Tilemap – klasa zajmująca się grafiką komórek i zmianą ich stanów, dziedzicząca z klas Drawable i Transformable z biblioteki SFML, pozwalające na rysowanie i przekształcanie obiektów. Posiada następujące metody:

- prywatne:
• *void Copy(*int* states, int* statescopy*, int width, int height)* – kopiuje wartości z tablicy stanów do pomocniczej tablicy przechowującej stany komórek z poprzedniej tury, przyjmuje odpowiednio wskaźniki do obu tablic i ilość komórek na szerokość oraz wysokość.
• *void Change(int i, int j, int* times, int* states, int prob)* – zaraża z pewnym prawdopodobieńswem komórkę, przyjmuje po kolei: indeksy i oraz j pozwalające na określenie pozycji w tablicy, wskaźniki do tablic czasów oraz prawdopodobieństwo zarażenia.

- pubiczne:
• *TileMap()* – konstruktor ustawiający początkowe wartości zmiennych
• *void load(const string& tileset, Vector2u tileSize, int width, int height)* - ustawia graficzny wygląd stanów komórek, pobiera referencję do stałego łańcucha znaków – nazwę pliku z którego pobierane są grafiki, Vector2u przechowujący wielkości komórek w pikselach na wysokość i szerokość, oraz ilość komórek na szerokość i wysokość.
• *void draw(RenderTarget& target, RenderStates states) const* – metoda rysująca grafiki komórek, pobiera referencje do podstawowego elementu klasy SFML, oraz tablicę stanów.
• *void Cells(Vector2u tileSize, int* states, int width, int height)* – metoda zmieniająca grafiki komórek, pobiera Vector2u przechowujący wielkość komórek w pikselach na szerokość i wysokość, wskaźnik do tablicy stanów oraz ilość komórek na szerokość i wysokość.
• *void States(int* times, int* states, int* statescopy, int width, int height, int sicktime, int immunitytime, int prob)* – zajmuje się najważniejszą mechaniką – dotyczącą zmian stanów komórek, pobiera wskaźniki do tablic czasów, stanów i pomocniczej stanów, ilość komórek na wysokość i szerokość, czas chorowania, odporności i prawdopodobieństwo zachorowania.
Ponadto klasa Tilemap posiada następujące zmienne wykorzystywane w metodach:

- prywatną:
• *chance* – zmienna przechowująca pseudolosową liczbę, pomaga w losowym zarażeniu komórki zdrowej przez sąsiednią chorą wylosowaną.

- publiczne:
• *VertexArray m_vertives* – obiekt przechowujący w tym wypadku czworokąt, jako rodzaj rysowalnego obiektu z biblioteki SFML, ilość jego wierzchołków, oraz rozmiar komórek.
• *Texture m_tileset* – przechowuje tekstury komórek pobrane z pliku.
• *int sicktime* – zmienna przechowująca ilość jednostek czasu, przez jaką zarażona komórka będzie chora.
• *int immunitytime* – zmienna przechowująca ilość jednostek czasu, przez jaką wyleczona komórka będzie odporna.
• *int prob* – zmienna przechowująca szanse na zarażenie komórki zdrowej przez sąsiednią chorą, modyfikowana przez dwa przyciski z klasy *Liszaj: probability, probability2.*
• *width* – zmienna przechowująca ilość komórek na szerokość.
• *heigth* – zmienna przechowująca ilość komórek na wysokość.
• *Xsize* – zmienna przechowująca wielkość w pikselach komórek na szerokość.
• *Ysize* – zmienna przechowująca wielkość w pikselach komórek na wysokość.

Buttons – klasa zajmująca się przyciskami i napisami na nich w oknie dostępnymi dzięki bibliotece SFML. Posiada następujące metody:
- publiczne:
• *Buttons(int x, int y, int width, int heigth, Font *font, string text, Color nColor, int size)* – konstruktor tworzący przyciski. Pobiera pozycję przycisku na osi x i y, jego szerokość i wysokość, czcionkę napisu, napis, jaki ma się na nim znajdować, kolor przycisku oraz wielkość czcionki napisu.
• *bool isOver(RenderWindow &window)* - metoda sprawdzająca, czy myszka jest nad przyciskiem, pobiera referencje do okna.
• *void Render(RenderTarget *target)* – metoda rysująca przyciski i napisy na nich, pobiera wskaźnik do podstawowego obiektu klasy SFML, tutaj jest nim zawsze okno.
• *bool change()* – zmienia stan przycisku z aktywny na nieaktywny i przeciwnie, używana przez przycisk zatrzymujący i wznawiający symulację.
• *void Resize(int xsize, int ysize, int x, int y, int width, int heigth, int size)* – zmienia rozmiary przycisków I napisów znajdujących się na nich, wywoływana przy zmianie wielkości okna. Pobiera wielkość okna na szerokość oraz wysokość, pozycje na osi x i y przycisku, wielkość przycisku na szerokość i wysokość, oraz wielkość napisu.

Ponadto klasa Buttons posiada następujące zmienne wykorzystywane w metodach:
- prywatne:
• *short unsigned buttonState* – zmienna przechowująca stan przycisku, wykorzystywana do zatrzymywania i wznawiania symulacji przyciskiem Start/Stop. Przyjmuje jedną z wartości unactive/active z typu wileczeniowego enum, tutaj globalne *button_states*.
• *RectangleShape shape* – obiekt klasy RectangleShape z biblioteki SFML, jest to prostokąt reprezentujący przycisk w oknie.
• *Font *font* – wskaźnik na czcionkę.
• *Text text* – obiekt klasy Text z biblioteki SFML, przechowuje napis na przycisku i jego właściwości.
• *Color nColor* – obiekt klasy Color z biblioteki SFML, przechowuje kolor przycisku.

Strings – klasa zajmująca się napisami w oknie dostępnymi dzięki bibliotece SFML. Posiada następujące metody:

- publiczne:
• Strings(int x, int y, Font *font, string text, int size) – konstruktor tworzący napis. Pobiera pozycję x I y napisu, wskaźnik do czcionki, łańcuch znaków i rozmier czcionki.
• void Render(RenderTarget *target) - metoda wypisująca tekst, pobiera wskaźnik do podstawowego obiektu klasy SFML, tutaj jest nim zawsze okno.
• void ChangeText(string text) – zmienia łańcuch w napisie, wykorzystywana po zmianach dodatkowych właściwości symulacji.
• void Resize(int xsize, int ysize, int x, int y, int size) – zmienia rozmiary napisów, wywoływana przy zmianie wielkości okna. Pobiera wielkość okna na szerokość oraz wysokość, pozycje na osi x i y napisu, oraz wielkość czcionki.

Ponadto klasa Strings posiada następujące zmienne wykorzystywane w metodach:
- prywatne:
• Font* font - wskaźnik na obiekt Font z biblioteki SFML, czyli na czcionkę
• Text text - obiekt Text z biblioteki, przechowuje łańcuch i jego właściwości.

Main() – ta funkcja jedynie tworzy obiekt klasy Liszaj, ustawia punkt startowy do generowania liczb pseudolosowych i uruchamia główną pętlę symulacji znajdującą się w metodzie Start klasy Liszaj.