# Министерство образования и науки РФ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Омский государственный технический университет»

|  |  |
| --- | --- |
| Факультет (институт) | Факультет информационных технологий и компьютерных систем |
|  |  |
| Кафедра | Информатики и вычислительной техники |
|  |  |

КУРСОВОЙ ПРОЕКТ (РАБОТА)

|  |  |
| --- | --- |
| по дисциплине | Операционные системы |
|  |  |
| на тему | Многопоточная Linux графическая модель транспортных перевозок дилижансами по двум круговым маршрутам между четырьмя городами. |

Пояснительная записка

|  |  |
| --- | --- |
| Шифр проекта (работы) | 020-КП-09.03.01-01-ПЗ |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | Студента (ки) | | | | Зайцев Владимир Александрович | | | | | | | | | | | |  |
|  |  |  |  | | | | фамилия, имя, отчество полностью | | | | | | | | | | | |  |
|  |  |  | Курс | 2 |  | | | Группа | | | | ИВТ-172 | | | |  | | |  |
|  |  |  |  |  | |  | | | |  |  | | | |  | |  | | |
|  |  |  | Направление (специальность) | | | | | | | | |  | | | | | | |  |
|  |  |  | ***09.03.01 (правильный номер)*** | | | | | | | | | ***Информатика и вычислительная техника*** | | | | | | |  |
|  |  |  | код, наименование | | | | | | | | | | | | | | |  | |
|  |  |  | Руководитель | | | | доцент, к.т.н | | | | | | | | | | | |  |
|  |  |  |  | | | | ученая степень, звание | | | | | | | | | | | |  |
|  |  |  | Флоренсов Александр Николаевич | | | | | | | | | | | | | | |  | |
|  |  |  | фамилия, инициалы | | | | | | | | | | | | | | |  | |
|  |  |  | Выполнил (а) | | | |  | | | | | | | | | | | |  |
|  |  |  |  | | | | дата, подпись студента (ки) | | | | | | | | | | | |  |
|  |  |  | К защите | | | |  | | | | | | | | | | | |  |
|  |  |  |  | | | | дата, подпись руководителя | | | | | | | | | | | |  |
|  |  |  | Выполнение и подготовка к защите КП (КР) | | | | | | Защита КП (КР) | | | | | Итоговый рейтинг | | | | | |
|  |  |  |  | | | | | |  | | | | |  | | | | | |
|  |  |  | Проект (работа) защищен (а) с оценкой | | | | | | | | | |  | | | | |  | |

Омск 2019

Министерство образования и науки РФ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Омский государственный технический университет»

**ОТЗЫВ**

**на курсовой проект (работу)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Факультет (институт) | | | | | | | | Факультет информационных технологий и компьютерных систем | | | | | | | | | | | | | |  | |
|  | | | | | | | |  | | | | | | | | | | | | | |  | |
| Кафедра | | | Информатики и вычислительной техники | | | | | | | | | | | | | | | | | |  | | |
|  | | |  | | | | | | | | | | | | | | | | | |  | | |
| Дисциплина | | | | Операционные системы | | | | | | | | | | | | | | | |  | | | |
|  | | | |  | | | | | | | | | | | | | | | |  | | | |
| Тема | Многопоточная Linux графическая модель транспортных перевозок дилижансами по двум круговым маршрутам между четырьмя городами. | | | | | | | | | | | | | | | | | | | | | |  |
|  | |  | | | | | | | | | | | | | | | | | | |  | | |
| Студент (ка) | | | | Зайцев Владимир Александрович | | | | | | | | | | | | | | | |  | | | |
|  | | | | фамилия, имя, отчество полностью | | | | | | | | | | | | | | | |  | | | |
| Курс | 2 | | | | Группа | | | | ИВТ-172 | |  | | | | | | | | | |  | | |
|  | |  | | | |  | | | |  | |  | | | | | | | | | |  | |
| Руководитель | | | | | Флоренсов Александр Николаевич | | | | | | | | | | | | | | | | |  | |
|  | | | | | ученая степень, звание, ФИО | | | | | | | | | | | | | | | | |  | |
|  | | | | | | | | | | | | | | | | | | | | |  | | |
| **Содержание отзыва** | | | | | | | | | | | | | | | | | | | | |  | | |
|  | | | | | | | | | | | | | | | | | | | | |  | | |
|  | | | | | | | | | | | | | | | | | | | | |  | | |
|  | | | | | | | | | | | | | | | | | | | | |  | | |
|  | | | | | | | | | | | | | | | | | | | | |  | | |
|  | | | | | | | | | | | | | | | | | | | | |  | | |
|  | | | | | | | | | | | | | | | | | | | | |  | | |
|  | | | | | | | | | | | | | | | | | | | | |  | | |
|  | | | | | | | | | | | | | | | | | | | | |  | | |
| **Рейтинговые баллы за выполнение и подготовку к защите курсового проекта (работы)** | | | | | | | | | | | | | | | |  | | | | | | | |
| **Заключение о допуске к защите** | | | | | | | | | | | | |  | | | | | | | | | | |
| **Руководитель** | | | | | | |  | | | | | | | Дата |  | | 20 |  | г. | | |  | |

Реферат

Пояснительная записка по курсовому проекту 25 с., 4 ч., 7 рис., 3 источ, 1 прил.

C, WINDOWS, GDI, GCC, МНОГОПОТОЧНОЕ ПРИЛОЖЕНИЕ.

Объектом исследования является организация взаимодействя нескольких потоков в операционной системе Windows при работе в графическом окне.

Цель работы – разработка многопоточной графической модели имитации рабочих поездок пассажиров по круговому чартерному международному авиамаршруту.

В результате работы разработан алгоритм взаимодействия нескольких нитей между собой и отоборажения результатов их работы в графическом окне в ОС Windows. После чего была разработана сама программа, имитирующая работу одного самолета на круговом маршруте, а так же симулирующая поведение людей пользующихся этим самолетом. Все это отображается в отдельном графическом окне операционной системы Windows.

Содержание

[Введение 4](#_Toc512456690)

[1 Введение в проблематику разработки многопоточных приложений 5](#_Toc512456691)

[2 Декомпозиция разрабатываемой программы снизу-вверх с формированием основных процедур ее функционирования и описанием их функционального назначения 6](#_Toc512456692)

[3 Описание глобальных информационных объектов программы: глобальных переменных, средств синхронизации потоков и используемых структур данных в случае их применения](#_Toc512456693) 7

[4 Детальное текстовое описание на основе сочетания естественного языка и программных конструкций алгоритмов всех процедур](#_Toc512456694) 8

[Заключение](#_Toc512456695) 14

[Список использованных источников](#_Toc512456696) 15

[Приложение](#_Toc512456697) 16

Введение

Курсовой проект по дисциплине «Операционные системы», 2 курс. В проекте использовался язык программирования C.

Задача: Разработать для Windows многопоточную программу, которая имитирует рабочие поездки пассажиров по круговому чартерному международному авиамаршруту Нью-Йорк-Лондон — Москва — Пескн. На маршруте летает один самолет, вмещающий до 4 пассажиров. При запуске программы, по запросу к пользователю, задается значение параметра М — суммарное число пассажиров, использующих указанные маршруты. Для каждого пассажира, который должен моделироваться отдельной нитью, случайным образом задается время пребывания в городе, после истеченя которого он появляется в аэропорту и ожидает нужный ему самолет, после прилета пассажир случайное время остается в гроде, затем выбирает новый город для поездки и этот процесс повторяется. Выбор направленя чартерного маршрута осуществляется случайным образом в момент появления пассажира в аэропорту. Первоначально все М пассажиров появляются в Москве. Отображение динамики модели должно осуществляться в графическом окне с помощью условных схем, значков и текстов, размер окна не менее 600 на 800 пикселей. Изображение городов и дорог должны быть представлены с помощью специально подобранных символов и схем. Рекомендуется разместить изображения городов в различных углах окна. Поведение каждого пассажира дожно имитироваться с помощью отдельной нити. Для правильного взаимодействия использовать семафоры или мьютексы.

Проект состоит из четырех разделов:

* Введение в проблематику разработки многопоточных приложений
* Декомпозиция разрабатываемой программы снизу-вверх с формированием основных процедур ее функционирования и описанием их функционального назначения
* Описание глобальных информационных объектов программы: глобальных переменных, средств синхронизации потоков и используемых структур данных в случае их применения
* Детальное текстовое описание на основе сочетания естественного языка и программных конструкций алгоритмов всех процедур

1 Введение в проблематику разработки многопоточных приложений

При разработке многопоточных приложений возникает ряд проблем.

В первую очередь нужно помнить, что в зависимости от системы при компиляции может оказаться необходимым явное указание того, что данное приложение будет использовать более одной нити. Тем не менее в операционной системе Windows этот вопрос не рассматривается, т.к. там все есть по умолчанию.

При компиляции проекта можно подключить библиотеки, необходимые для работы с консольными окнами в Windows. Так, для данного курсового проекта оказалось необходимым использование команд –lgdi32 для рисования.

В современных операционных системах широко используются потоки (thread). Именно они применяются и в данном курсовом проекте. Обычно в потоке выполняются действия одной из процедур программы. Хоть любой нити процесса и доступны все части программы этого процесса, реально же работа организуется так, чтобы нити отвечала отдельная процедура. Учитывая, что процедуре для нормальной работы необходимы локальные переменные, становится понятным закрепление области этих переменных за нитью. Объект хранения локальных переменных (вместе со служебной информацией при вызове подпрограмм) называют стеком. Этот стек в действительности является частью оперативной памяти, он используется не только программно, но и аппаратно.

Так же следует уделить внимание организации взаимодействия различных потоков друг с другом, а так же с памятью, к которой эти потоки могут обращаться одновременно. Дело в том, что нередко могут возникать такие ситуации, когда два или более потоков пытаются модифицировать одну и ту же область памяти. В таких случаях возникает так называемая ситуация состязания нитей, когда результаты работы их обоих перемешиваются, что в результате приводит к неопределенному поведению программы, различным ошибкам и прочим нежелательным последствиям. Одним из удобных средств для синхронизации работы нескольких потоков являются мьютексы. Их задача состоит в организации одновременного доступа к области памяти не более одного потока.

2 Декомпозиция разрабатываемой программы снизу-вверх с формированием основных процедур ее функционирования и описанием их функционального назначения

Поставленная задача была разделена в целях упрощения разработки и в результате были выделены следующие подзадачи:

-Алгоритм симулирования жителей пользующихся самолетом для перемещения между городами.

-Алгоритм симулированя перемещения саммолета между городами с остановками в аэропорттах каждого города.

-Алгоритм отображения ддинамики модели в графическом окне операционной системы.

-Первоначальная настройка необходимого окружения при запуске программы, настройка окна, создание пассажиров, запуск процессов.

Алгоритм поведения пассажиров находится в функции passajirFunction. Он предназначен для симуляции каждого отдельного пассажира. При этом данная фнукция выполняется многими запущенными прроцессами, каждый из которых по отдельности симулирует поведение одного пассажира.

Графическое отображение производится в функции paintFunction, с помощью графических функций из встроенного в windows.h функционала gdi (с помощью точек, многоугольниых фигур и текста).

Алгоритм перемещений самолета было решено поместить в бесконечный цикл в основной поток в фугкцию main, т. к. по заданию он всего один.

Первоначальная настройка всех необходимых компонентов для отображения и для реализации многопоточности происходит так же в функции main при старте программы перед входом в бесконечный цикл отрисовки. Там создается окно программы, создаются отдельные пассажиры, создаются потоки, в которые передаетсся функция управления пассажиром, и аргумент, указывающий, где именно в общем массиве всех пассажиров находится тот, которым должен управлять конкретный, создаваемый поток.

3 Описание глобальных информационных объектов программы: глобальных переменных, средств синхронизации потоков и используемых структур данных в случае их применения

В программе используются следующие глобальные переменные:

1. hstdout — хендл стандартного вывода
2. hInstance — переменная содержащая экзепляр созданного окна
3. si — переменная содержащая информацию о старте окна
4. hwnd — дескриптор окна
5. hdc — дескриптор для отображения графики в окне
6. msg — переменная храянящая последнее переданное окном сообщение
7. pasChisloC — массив из четырех символов, куда записывается вводимое число пассажиров
8. pasChislo — в эту переменную сохранятеся число пассажиров преобразованное из строки из предыдущей переменной
9. curDigit — номер текущей вводимой цифры
10. brushGreen — brushBlue — кисти разных цветов для закрашивания областей
11. pen — перменная содержащая информацию о том, как нужно рисовать линии
12. samoletForma — массив содержащий координаты вершин многоугольника отображающего самолет
13. samoletPosRis — координаты самолета в целых числах
14. samoletPosX, samoletPosY — координаты в вещенственных числах
15. samoletReis — кордината точки в которую летит самолет в данный момент
16. samoletStop, samoletPolet — текущее состояние самолета
17. samoletPasss — количество пассажиров в самолете
18. samowlrGorodTekush — номер города в котором сейчас находится самолет
19. gorod1Pos — gorod4Pos — позиции городов
20. oblastForma, gorodForma — многоугольники всех городов с разными формами
21. struct passajir — структура хранящая всю необходимую информацию о пассажире
22. passajiry — массив со всеми пассажирами
23. hmtx — мьютекс для ограничения доступа пассажиров к самолету в необходимый момент, чтобы в самолете не оказалось более 4 пассажиров.

4 Детальное текстовое описание на основе сочетания естественного языка и программных конструкций алгоритмов всех процедур

Функция содержащая алгоритм пассажиров работает по принципу изменения состояний. Каждый пассажир может находиться в одном из нескольких возможных состояний. Например он может спать, идти к аеропорту, лететь в самолете или идти от аэропорта в город. В функции происоходит бесконечный цикл, который при каждом своем проходе проверяет состояние пассажира, и в зависимости от этого может изменять его координаты, или его состояние. Эти данные постоянно берутся из ячейки массива номер которой равен переданному при создании потока аргументу arg. После произведения действий, они помещаются обратно в массив в ту же ячейку. В данной функции содержится область программы, в которой может произойти так называемая ситуация состязания процессов. Это место, где пассажиры пытаются сесть в самолет. Весь процесс посадки помещен между вызовами WaitForSingleObject и ReleaseMutex с параметром hmtx что гарантирует, что в один момент времени сможет сесть только один пассажир, а остальные будут ждать до тех пор, пока не получат доступ к данному мьютексу hmtx.

Алгоритм работы самолета похож на функцию пассажира. Он находится прямо внутри main, в бесконечном цикле. В нем так же при каждом проходе цикла происходит изменение данных описывающих самолет и его текущее состояние.

Отрисовка расположена в отдельной функции paintFunction, но тем не менее она так же как и алгоритм самолета вызывается в бесконечном цикле в main. Данная функция с помощью цикла for получает данные из массива с пассажирами и по ним рисует точки на экране. Так же она берт координаты самолета и координаты городов. Каждый раз отрисовывая все эти обьекты. Для создания иллюзии движения перед всеми этими функциями выполняется заполнение всего окна монтонным цветом, а сама функция выполняется приблизительно 30 раз в секунду.

Перед тем как войти в бесконечный цикл в функции main выполняется создание окна заданного размера. Создание мьютекса. Опрос пользователя о количесве создаваемых пассажиров. Создание пассажиров, создание потоков, и передача в потоки функции управления пассажирами и числа указывающего на ячейку в массиве хранящую информацию о пассажире который будет управляться данным потоком.

Схема алгоритма главной процедуры представлена на рисунке 1.

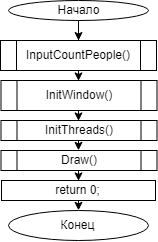


Рисунок 1 – схема алгоритма основной процедуры

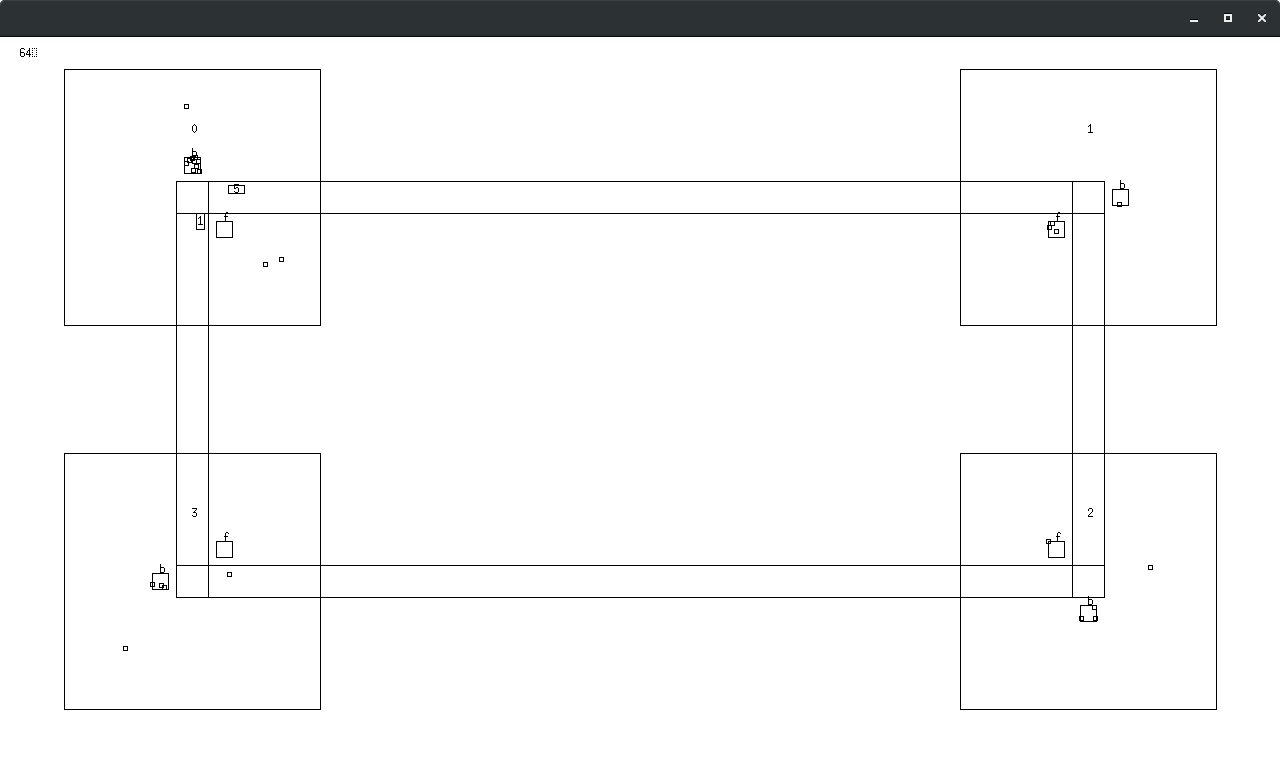
Скриншот работы программы представлен на рисунках 2-6.

Рисунок 2 – пример работы программы

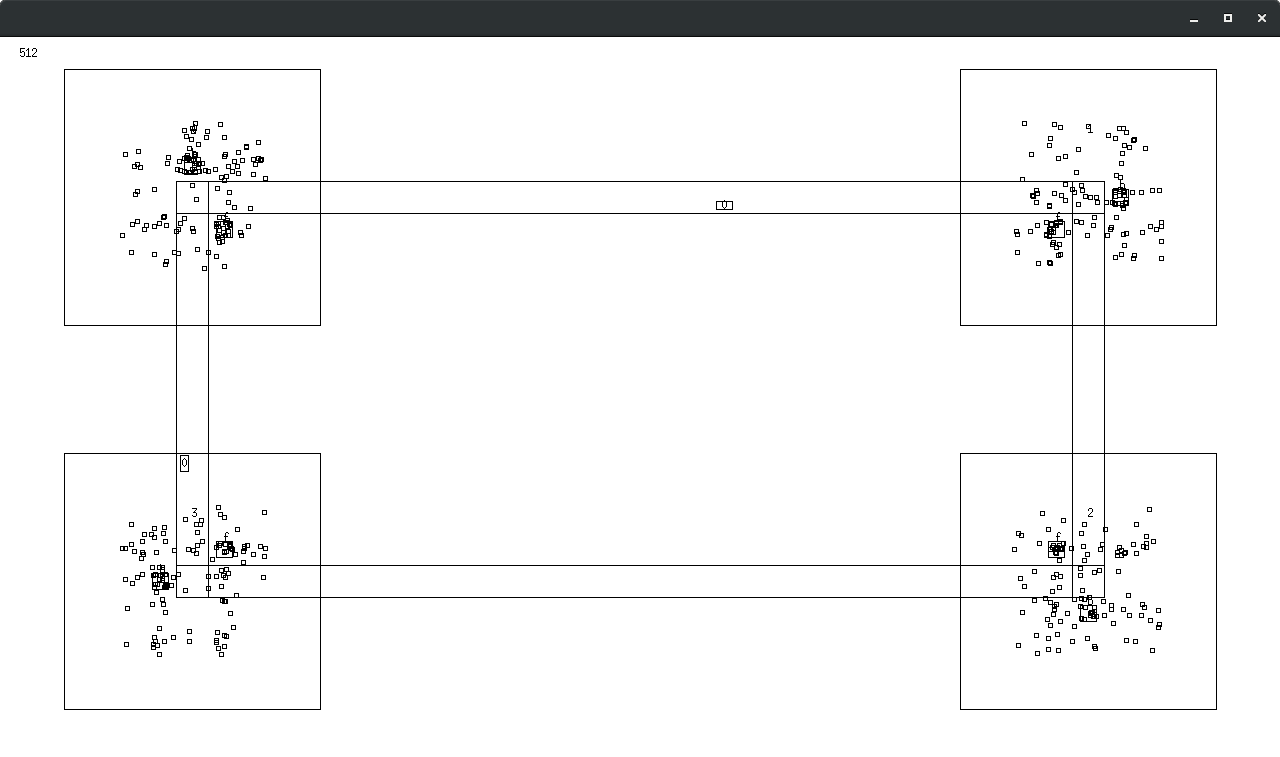


Рисунок 3 – пример работы программы

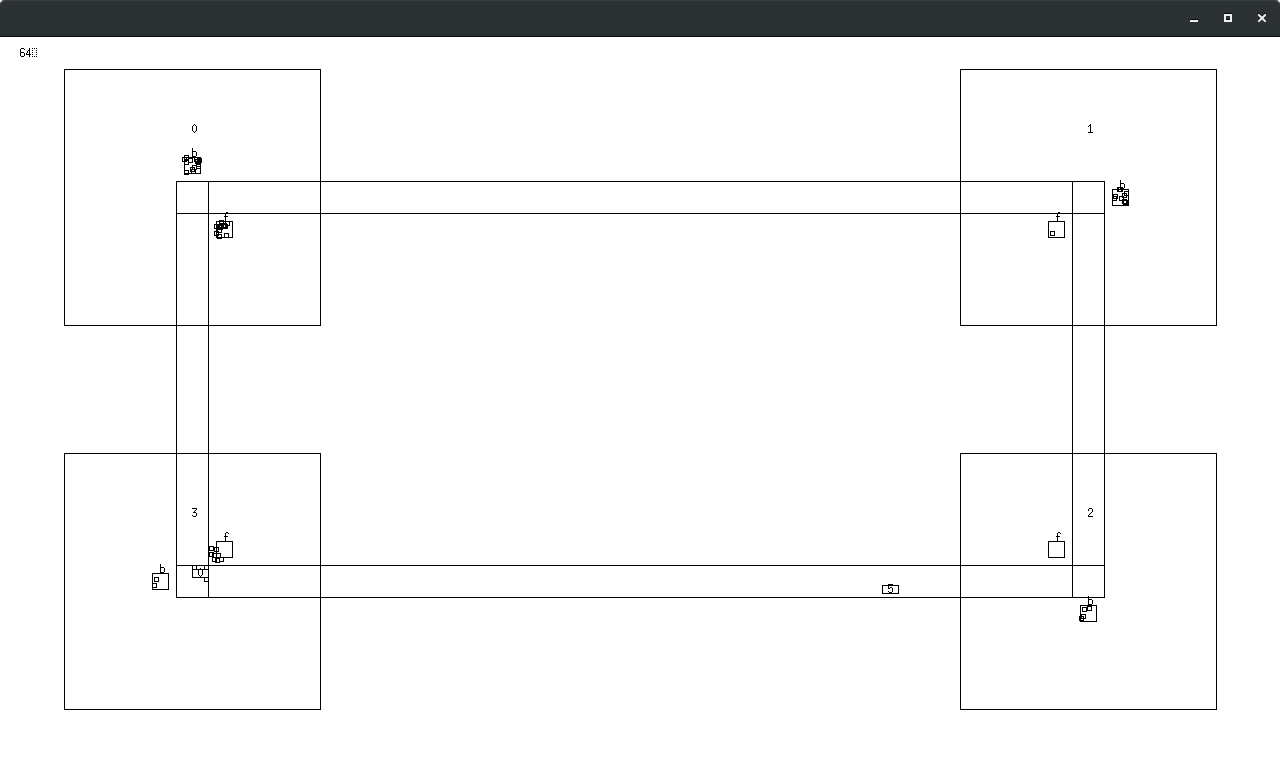


Рисунок 4 – пример работы программы

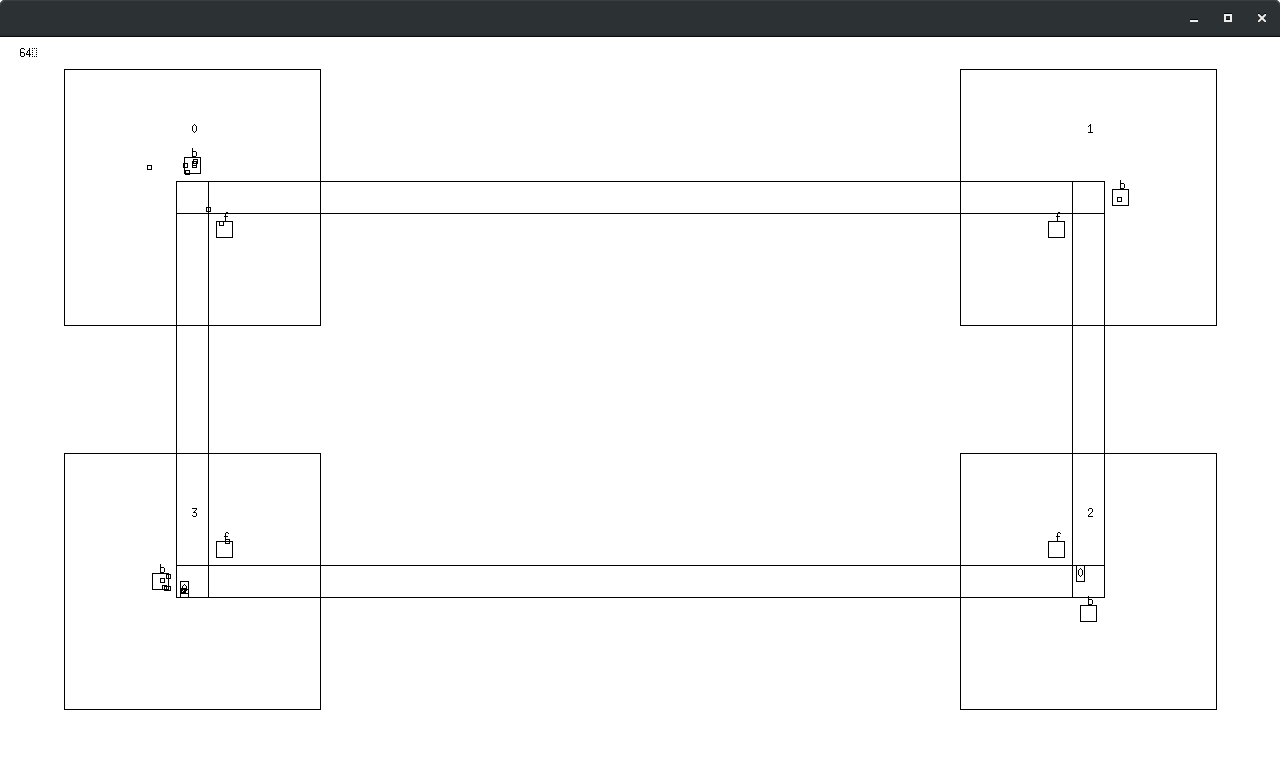
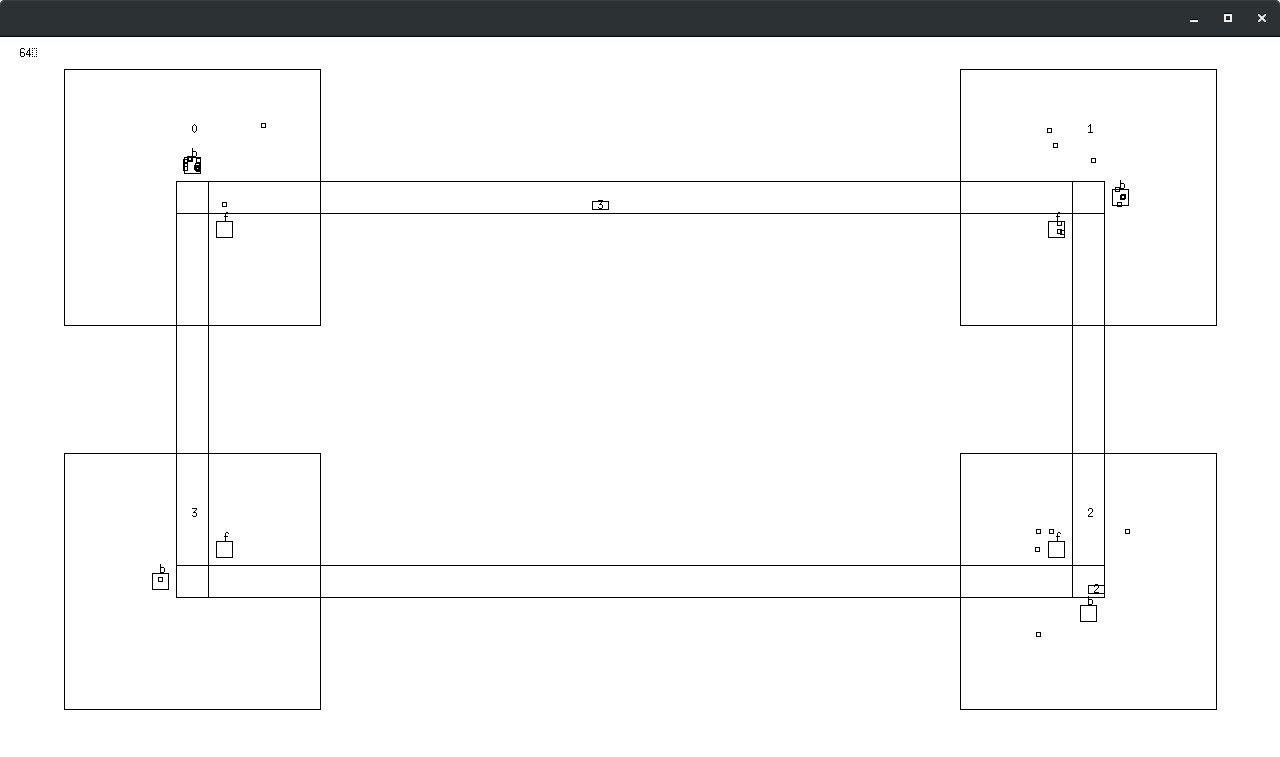
Рисунок 5 – пример работы программы

Рисунок 6 – пример работы программы

Скриншот файла для трансляции программы представлен на рисунке 7.

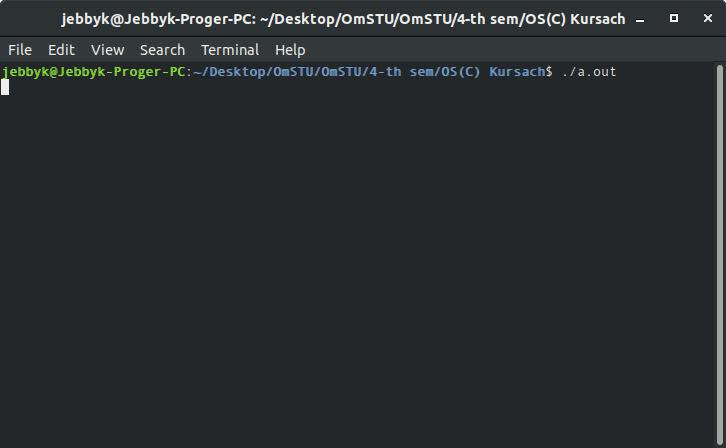


Рисунок 7 – cкриншот файла для трансляции программы

Заключение

В результате работы разработан алгоритм взаимодействия нескольких нитей между собой и отоборажения результатов их работы в графическом окне в ОС Windows. После чего была разработана сама программа, имитирующая работу одного самолета на круговом маршруте, а так же симулирующая поведение людей пользующихся этим самолетом. Все это отображается в отдельном графическом окне операционной системы Windows.

Список использованных источников

1. Флоренсов, А.Н. Операционные системы для программиста. Омск. ОмГТУ, 2005.
2. Гордеев, А.В. Операционные системы / А.В. Гордеев. – 2-е изд. – СПб.: Питер, 2007
3. ГОСТ 7.1-2003 Библиографическая запись. Библиографическое описание. Общие требования и правила составления.

Приложение

**Исходный код программы**

|  |  |
| --- | --- |
|  | #include <windows.h> |
|  | #include <stdio.h> |
|  | #include <math.h> |
|  |  |
|  | HANDLE hstdout;//хендл стандартного вывода |
|  |  |
|  | HINSTANCE hInstance;//для создания окна |
|  | STARTUPINFO si;//информация о старте окна |
|  | int nCmdShow; |
|  | HWND hwnd;//дескриптор окна |
|  | HDC hdc;//дескриптор для рисования |
|  | MSG msg;//сообщение передаваемое окном |
|  | WNDCLASS wc;//для создания окна |
|  | char passChisloC[4] = "";//строка куда вводится число пассажиров |
|  | int passChislo = 1;//число в котором преобразуется число пасиков из строки в цифру |
|  | int curDigit = 0;//для вода числа пасиков тукущий вводимый разряд числа) |
|  |  |
|  | HBRUSH brushGreen;//кисти для заливки разными цветами |
|  | HBRUSH brushGray; |
|  | HBRUSH brushBlue; |
|  | HBRUSH brushBlack; |
|  | HBRUSH brushLightBlue; |
|  | HBRUSH brushWhite; |
|  |  |
|  | HPEN pen;//карандаш для рисования линий |
|  |  |
|  | POINT samoletForma[] = {//координаты точек по которым строится форма самолета |
|  | {0,24}, |
|  | {2, 22}, |
|  | {4, 5}, |
|  | {20,-2}, |
|  | {20,-4}, |
|  | {4, -2}, |
|  | {2, -20}, |
|  | {-2, -20}, |
|  | {-4, -2}, |
|  | {-20,-4}, |
|  | {-20,-2}, |
|  | {-4, 5}, |
|  | {-2, 22} |
|  | }; |
|  | POINT samoletPosRis;//позиция самолета для рисования |
|  | float samoletPosX;//уточненная позиция самолета(нужно для нормлаьного перемещения) |
|  | float samoletPosY; |
|  |  |
|  | POINT samoletReis;//позиция куда самолету надо лететь |
|  | BOOL samoletStop;//стоит ли самолет на стоянке |
|  | BOOL samoletPolet;//или же летит |
|  | int samoletPass;//число пасиков в самолете |
|  | int samoletGorodTekush = 0;//текущий город в котром находится самолет |
|  | int samoletSon = 0;//сон самолета(для ожидания) |
|  | float samoletScorost = 0;//скорость самолета |
|  |  |
|  | int tScorost = 30;//скорость и плавность анимации |
|  |  |
|  | char text[12] = ""; |
|  |  |
|  |  |
|  | POINT gorod1Pos = { 128, 128};//позиции городов |
|  | POINT gorod2Pos = { 620, 200}; |
|  | POINT gorod3Pos = { 500, 600}; |
|  | POINT gorod4Pos = { 200, 580}; |
|  |  |
|  | POINT oblast1Forma[] = {//координаты с формами для зеленых и серых областей |
|  | {-100, 0}, |
|  | {-80, -20}, |
|  | {-30, -25}, |
|  | {-30, -60}, |
|  | {0, -100}, |
|  | {20, -50}, |
|  | {60, -70}, |
|  | {60, -20}, |
|  | {80, 0}, |
|  | {70, 20}, |
|  | {40, 40}, |
|  | {0, 80}, |
|  | {-20, 100}, |
|  | {-30, 50}, |
|  | {-80, 10} |
|  | }; |
|  | POINT gorod1Forma[] = { |
|  | {-30, 40}, |
|  | {-20, 46}, |
|  | {10, 42}, |
|  | {14, 32}, |
|  | {22, 32}, |
|  | {24, 38}, |
|  | {44, 18}, |
|  | {44, 6}, |
|  | {18, 8}, |
|  | {18, -8}, |
|  | {6, -18}, |
|  | {0, -20}, |
|  | {-10, 16}, |
|  | {24, -20}, |
|  | {-34, 0}, |
|  | {-20, 12}, |
|  | {-22, 20}, |
|  | {-36, 24}, |
|  | {-44, 26}, |
|  | {-44, 34} |
|  | }; |
|  |  |
|  | POINT oblast2Forma[] = { |
|  | {7,168}, |
|  | {-32,175}, |
|  | {-55, 148}, |
|  | {-90,139}, |
|  | {-100,100}, |
|  | {-85,80}, |
|  | {-83, 52}, |
|  | {-93,22}, |
|  | {-65,-11}, |
|  | {-59,-26}, |
|  | {-62,67}, |
|  | {-67,-92}, |
|  | {-17,-107}, |
|  | {20, -75}, |
|  | {56, -89}, |
|  | {58, -40}, |
|  | {33, -21}, |
|  | {42, 14}, |
|  | {60, 31}, |
|  | {58, 86}, |
|  | {34,107}, |
|  | {25, 140}, |
|  | {56, 166}, |
|  | {26, 180} |
|  | }; |
|  | POINT gorod2Forma[] = { |
|  | {-30, 40}, |
|  | {-20, 46}, |
|  | {10, 42}, |
|  | {14, 32}, |
|  | {22, 32}, |
|  | {24, 38}, |
|  | {44, 18}, |
|  | {44, 6}, |
|  | {18, 8}, |
|  | {18, -8}, |
|  | {6, -18}, |
|  | {0, -20}, |
|  | {-10, 16}, |
|  | {24, -20}, |
|  | {-34, 0}, |
|  | {-20, 12}, |
|  | {-22, 20}, |
|  | {-36, 24}, |
|  | {-44, 26}, |
|  | {-44, 34} |
|  | }; |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | POINT oblast3Forma[] = { |
|  | {-40, 45}, |
|  | {-42, 91}, |
|  | {-2, 91}, |
|  | {42, 44}, |
|  | {93, 38}, |
|  | {120, -27}, |
|  | {104, -65}, |
|  | {99, -110}, |
|  | {74, -108}, |
|  | {70, -70}, |
|  | {47, -33}, |
|  | {-16, -48}, |
|  | {-41, -35}, |
|  | {-74, 21} |
|  | }; |
|  | POINT gorod3Forma[] = { |
|  | {-30, 40}, |
|  | {-20, 46}, |
|  | {10, 42}, |
|  | {14, 32}, |
|  | {22, 32}, |
|  | {24, 38}, |
|  | {44, 18}, |
|  | {44, 6}, |
|  | {18, 8}, |
|  | {18, -8}, |
|  | {6, -18}, |
|  | {0, -20}, |
|  | {-10, 16}, |
|  | {24, -20}, |
|  | {-34, 0}, |
|  | {-20, 12}, |
|  | {-22, 20}, |
|  | {-36, 24}, |
|  | {-44, 26}, |
|  | {-44, 34} |
|  | }; |
|  |  |
|  | POINT oblast4Forma[] = { |
|  | {-67, 27}, |
|  | {-29, 28}, |
|  | {-65, 81}, |
|  | {-22, 57}, |
|  | {6, 80}, |
|  | {35, 62}, |
|  | {36, 22}, |
|  | {84, 16}, |
|  | {46, -20}, |
|  | {62, -36}, |
|  | {80, -80}, |
|  | {40, -80}, |
|  | {16, -46}, |
|  | {-20, -44}, |
|  | {-40, -80}, |
|  | {-80, -82}, |
|  | { -100, -56}, |
|  | {-100, -16}, |
|  | {-94, 20}, |
|  | {-82, 68} |
|  | }; |
|  | POINT gorod4Forma[] = { |
|  | {-30, 40}, |
|  | {-20, 46}, |
|  | {10, 42}, |
|  | {14, 32}, |
|  | {22, 32}, |
|  | {24, 38}, |
|  | {44, 18}, |
|  | {44, 6}, |
|  | {18, 8}, |
|  | {18, -8}, |
|  | {6, -18}, |
|  | {0, -20}, |
|  | {-10, 16}, |
|  | {24, -20}, |
|  | {-34, 0}, |
|  | {-20, 12}, |
|  | {-22, 20}, |
|  | {-36, 24}, |
|  | {-44, 26}, |
|  | {-44, 34} |
|  | }; |
|  |  |
|  |  |
|  |  |
|  | HRGN oblast1;//переменные которые буду хранить неподсредственно фигуру областей |
|  | HRGN gorod1; |
|  | HRGN oblast2; |
|  | HRGN gorod2; |
|  | HRGN oblast3; |
|  | HRGN gorod3; |
|  | HRGN oblast4; |
|  | HRGN gorod4; |
|  |  |
|  | HANDLE hmtx; |
|  |  |
|  |  |
|  |  |
|  | typedef struct {//структура с данными об отдельном пассажире |
|  | float fx;//точные координаты (тоже для перемещения) |
|  | float fy; |
|  | int x;//целочисленные координаты для риосвания |
|  | int y; |
|  | int cx; |
|  | int cy; |
|  | BOOL walkAeroport;//к аеропорту |
|  | BOOL walkGorod;//к гоороду |
|  | BOOL samolet;//в самолете |
|  | BOOL ojidanie;// ждет самолета |
|  | BOOL son;//спит в городе |
|  | int sonDlinna;//число сколько еще спать |
|  | int gorodTekush;//городо в котором пассажир находится |
|  | }passajir; |
|  |  |
|  | passajir passajiry[1024];//массив с пассажирами |
|  |  |
|  | void passajirFunction(void \*arg)//функция пассажира (с ней тред запускается) |
|  | { |
|  | srand(arg); |
|  | int i = (int)arg;//передаем аргумент, по которому определяем какой именно элемент массива модифицеровать |
|  | passajiry[i].cx = gorod1Pos.x;//указываем начальные целевые координаты |
|  | passajiry[i].cy = gorod1Pos.y; |
|  | passajiry[i].gorodTekush = 0;//город |
|  | passajiry[i].walkAeroport = TRUE;//и начальное состояние, что он идет в аеропорт |
|  |  |
|  |  |
|  | while(TRUE) |
|  | { |
|  | if(passajiry[i].walkAeroport == TRUE)//если пассажир сейчас идет в аеропрт |
|  | { |
|  | float dx; |
|  | float dy; |
|  | dx = passajiry[i].cx - passajiry[i].fx;//берем вектор по направлению к целевой точке |
|  | dy = passajiry[i].cy - passajiry[i].fy; |
|  |  |
|  | float l = sqrt(dx\*dx + dy\*dy);// узнаем рассатояние |
|  | if(l > 1.0f) |
|  | { |
|  | dx /= l;//нормализуем вектор |
|  | dy /= l; |
|  | } |
|  |  |
|  | passajiry[i].fx += dx;//перемещаем человека |
|  | passajiry[i].fy += dy; |
|  | passajiry[i].x = passajiry[i].fx; |
|  | passajiry[i].y = passajiry[i].fy; |
|  |  |
|  | if(passajiry[i].fx > passajiry[i].cx -1 && passajiry[i].fx < passajiry[i].cx + 1 && passajiry[i].fy > passajiry[i].cy -1 && passajiry[i].fy < passajiry[i].cy + 1) |
|  | { |
|  | passajiry[i].walkAeroport = FALSE;//меняем его состояние на ожидание, если он очень близко к целевой точке |
|  | passajiry[i].ojidanie = TRUE; |
|  | } |
|  |  |
|  | } |
|  | if(passajiry[i].ojidanie == TRUE)//если сейчас ожидание |
|  | { |
|  | float dx; |
|  | float dy; |
|  | dx = samoletPosX - passajiry[i].fx; |
|  | dy = samoletPosY - passajiry[i].fy; |
|  | float l = sqrt(dx\*dx + dy\*dy);//так же беерм вектор до самолета и узнаем расстояние |
|  | WaitForSingleObject(hmtx, INFINITE); |
|  | if(l < 4.0f && samoletPass < 4 && samoletStop == TRUE)//если оно очень маленькое то пытаемся сесть в самолет обязательно блокируя всем остальным людям доступ к самолету |
|  | { |
|  |  |
|  | passajiry[i].ojidanie = FALSE; |
|  | passajiry[i].samolet = TRUE; |
|  | samoletPass++; |
|  | passajiry[i].sonDlinna = 200;//счетчик чтоб человек не спрыгивал с самолета когда не надо |
|  |  |
|  | } |
|  | ReleaseMutex(hmtx);//потом отпускаем мьютекс |
|  | } |
|  | if(passajiry[i].samolet == TRUE)//если пассажир в самолете |
|  | { |
|  | passajiry[i].sonDlinna--; |
|  | passajiry[i].fx = samoletPosX;//перемещаем человека вместе с самолетом |
|  | passajiry[i].fy = samoletPosY; |
|  | passajiry[i].x = passajiry[i].fx; |
|  | passajiry[i].y = passajiry[i].fy; |
|  | if(samoletStop == TRUE)//если самолет остановился |
|  | { |
|  | if(passajiry[i].sonDlinna <=0)// и уже можно слазить |
|  | { |
|  | passajiry[i].samolet = FALSE;//слазием с самолета |
|  | passajiry[i].gorodTekush = samoletGorodTekush; |
|  | passajiry[i].walkGorod = TRUE; |
|  |  |
|  |  |
|  | if(passajiry[i].gorodTekush == 0)//берем рандомную позицию по координатам города |
|  | { |
|  | passajiry[i].cx = (rand() % 100) + gorod1Pos.x - 50; |
|  | passajiry[i].cy = (rand() % 100) + gorod1Pos.y - 50; |
|  | } |
|  | if(passajiry[i].gorodTekush == 1) |
|  | { |
|  | passajiry[i].cx = (rand() % 100) + gorod2Pos.x - 50; |
|  | passajiry[i].cy = (rand() % 100) + gorod2Pos.y - 50; |
|  | } |
|  | if(passajiry[i].gorodTekush == 2) |
|  | { |
|  | passajiry[i].cx = (rand() % 100) + gorod3Pos.x - 50; |
|  | passajiry[i].cy = (rand() % 100) + gorod3Pos.y - 50; |
|  | } |
|  | if(passajiry[i].gorodTekush == 3) |
|  | { |
|  | passajiry[i].cx = (rand() % 100) + gorod4Pos.x - 50; |
|  | passajiry[i].cy = (rand() % 100) + gorod4Pos.y - 50; |
|  | } |
|  | } |
|  | } |
|  | } |
|  | if(passajiry[i].walkGorod == TRUE)//если идем в город |
|  | { |
|  | float dx; |
|  | float dy; |
|  | dx = passajiry[i].cx - passajiry[i].fx;//такое же перемещение по вектору до точки |
|  | dy = passajiry[i].cy - passajiry[i].fy; |
|  |  |
|  | float l = sqrt(dx\*dx + dy\*dy); |
|  | if(l > 1.0f) |
|  | { |
|  | dx /= l; |
|  | dy /= l; |
|  | } |
|  |  |
|  | passajiry[i].fx += dx; |
|  | passajiry[i].fy += dy; |
|  | passajiry[i].x = passajiry[i].fx; |
|  | passajiry[i].y = passajiry[i].fy; |
|  |  |
|  | if(passajiry[i].fx > passajiry[i].cx -1 && passajiry[i].fx < passajiry[i].cx + 1 && passajiry[i].fy > passajiry[i].cy -1 && passajiry[i].fy < passajiry[i].cy + 1) |
|  | { |
|  | passajiry[i].walkGorod = FALSE;//и если дошли, то засыпаем на долго |
|  | passajiry[i].son = TRUE; |
|  | passajiry[i].sonDlinna = rand() % 30000 + 30000; |
|  | Sleep(passajiry[i].sonDlinna); |
|  | passajiry[i].son = FALSE; |
|  | passajiry[i].walkAeroport = TRUE;//после того как проснулись, снова идем к аеропроту |
|  |  |
|  | if(passajiry[i].gorodTekush == 0) |
|  | { |
|  | passajiry[i].cx = gorod1Pos.x; |
|  | passajiry[i].cy = gorod1Pos.y; |
|  | } |
|  | if(passajiry[i].gorodTekush == 1) |
|  | { |
|  | passajiry[i].cx = gorod2Pos.x; |
|  | passajiry[i].cy = gorod2Pos.y; |
|  | } |
|  | if(passajiry[i].gorodTekush == 2) |
|  | { |
|  | passajiry[i].cx = gorod3Pos.x; |
|  | passajiry[i].cy = gorod3Pos.y; |
|  | } |
|  | if(passajiry[i].gorodTekush == 3) |
|  | { |
|  | passajiry[i].cx = gorod4Pos.x; |
|  | passajiry[i].cy = gorod4Pos.y; |
|  | } |
|  | } |
|  | } |
|  |  |
|  | Sleep(1000/tScorost);//задержка между отедльными кадрами анимации |
|  | } |
|  | } |
|  |  |
|  |  |
|  | void paintFunction()//фнукция рисования |
|  | { |
|  | Rectangle(hdc, 0,0,800,800);//монотонная заливка всего экрана |
|  |  |
|  |  |
|  | FillRgn(hdc, oblast1, brushGreen);//отрисоовка городов |
|  | FillRgn(hdc, gorod1, brushGray); |
|  |  |
|  | FillRgn(hdc, oblast2, brushGreen); |
|  | FillRgn(hdc, gorod2, brushGray); |
|  |  |
|  | FillRgn(hdc, oblast3, brushGreen); |
|  | FillRgn(hdc, gorod3, brushGray); |
|  |  |
|  | FillRgn(hdc, oblast4, brushGreen); |
|  | FillRgn(hdc, gorod4, brushGray); |
|  |  |
|  | POINT \_samoletForma[sizeof(samoletForma)/8];//берем форму самолета |
|  |  |
|  | for(int i = 0; i < sizeof(\_samoletForma)/8; i++)+//потом перемещаем ее в нужную позицию |
|  | { |
|  | \_samoletForma[i].x = samoletForma[i].x + samoletPosRis.x; |
|  | \_samoletForma[i].y = samoletForma[i].y + samoletPosRis.y; |
|  | } |
|  | HRGN samolet = CreatePolygonRgn(\_samoletForma, sizeof(samoletForma)/8,WINDING );//и заливаем |
|  | FillRgn(hdc, samolet, brushWhite); |
|  |  |
|  |  |
|  | for(int i = 0; i < passChislo; i++) |
|  | { |
|  | Rectangle(hdc, passajiry[i].x-1, passajiry[i].y-1, passajiry[i].x+1, passajiry[i].y+1);//рисуем маленьких человечков |
|  | } |
|  | } |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | void main() |
|  | { |
|  | hmtx = CreateMutex(NULL, FALSE, NULL);//главный и единственный мьютекс |
|  | LRESULT WINAPI WinProc(HWND hwnd, UINT tmsg, WPARAM wParam, LPARAM lParam)//создание окна и считывание его событий |
|  | { |
|  | switch(tmsg) |
|  | { |
|  | case WM\_DESTROY: |
|  | { |
|  | PostQuitMessage(0); |
|  | exit(0); |
|  | return 0; |
|  | } |
|  | case WM\_KEYDOWN: |
|  | { |
|  | hdc = GetDC(hwnd); |
|  | char key[1]; |
|  | key[0] = (char)wParam; |
|  | if(curDigit == -1) |
|  | { |
|  | exit(0); |
|  | } |
|  | if(key[0] >= '0' && key[0] <= '9') |
|  | { |
|  | TextOut(hdc, 128+64+16+curDigit\*8, 32, key, 1); |
|  | passChisloC[curDigit] = key[0]; |
|  | curDigit++; |
|  | }else{ |
|  | curDigit = -1; |
|  | } |
|  | return 0; |
|  | } |
|  | } |
|  | return DefWindowProc(hwnd, tmsg, wParam, lParam); |
|  | } |
|  | GetStartupInfo(&si); |
|  | if(si.dwFlags & STARTF\_USESHOWWINDOW) |
|  | { |
|  | nCmdShow = si.wShowWindow; |
|  | }else{ |
|  | nCmdShow = SW\_SHOWDEFAULT; |
|  | } |
|  | hInstance = GetModuleHandle(NULL); |
|  | memset(&wc, 0, sizeof(wc)); |
|  | wc.lpszClassName = "MyClass"; |
|  | wc.lpfnWndProc = WinProc; |
|  | wc.hCursor=LoadCursor(NULL, IDC\_ARROW); |
|  | wc.hbrBackground=(HBRUSH)(COLOR\_WINDOW+1); |
|  | wc.hInstance = hInstance; |
|  |  |
|  | if(!RegisterClass(&wc)) return; |
|  | hwnd = CreateWindow("MyClass", "OurWindow", WS\_OVERLAPPEDWINDOW, 0, 0, 800, 800, 0, 0, hInstance, NULL); |
|  | hdc = GetDC(hwnd); |
|  | ShowWindow(hwnd, nCmdShow);//тут заканчивается создание окна |
|  |  |
|  | brushGreen = CreateSolidBrush(RGB(64,128, 32));//создаем кисточки |
|  | brushBlue = CreateSolidBrush(RGB(32,128,255)); |
|  | brushGray = CreateSolidBrush(RGB(128,128,128)); |
|  | brushWhite = CreateSolidBrush(RGB(200,200,200)); |
|  | pen = CreatePen(PS\_SOLID, 1, RGB(0,0,0)); |
|  |  |
|  |  |
|  |  |
|  |  |
|  | char text1[] = "Vvedite chislo passajirov";//ввод числа человечков |
|  | TextOut(hdc, 32, 32, text1, sizeof(text1)); |
|  |  |
|  | while(GetMessage(&msg, 0, 0, 0) && curDigit > -1) |
|  | { |
|  | DispatchMessage(&msg); |
|  | } |
|  |  |
|  | sscanf(passChisloC, "%d", &passChislo);//преобразование полученной строки в цифру |
|  |  |
|  |  |
|  |  |
|  | for(int i = 0; i < sizeof(oblast1Forma)/8; i++)//расположение городов в нужных позициях |
|  | { |
|  | oblast1Forma[i].x += gorod1Pos.x; |
|  | oblast1Forma[i].y += gorod1Pos.y; |
|  | } |
|  | for(int i = 0; i < sizeof(gorod1Forma)/8; i++) |
|  | { |
|  | gorod1Forma[i].x += gorod1Pos.x; |
|  | gorod1Forma[i].y += gorod1Pos.y; |
|  | } |
|  | for(int i = 0; i < sizeof(oblast2Forma)/8; i++) |
|  | { |
|  | oblast2Forma[i].x += gorod2Pos.x; |
|  | oblast2Forma[i].y += gorod2Pos.y; |
|  | } |
|  | for(int i = 0; i < sizeof(gorod2Forma)/8; i++) |
|  | { |
|  | gorod2Forma[i].x += gorod2Pos.x; |
|  | gorod2Forma[i].y += gorod2Pos.y; |
|  | } |
|  | for(int i = 0; i < sizeof(oblast3Forma)/8; i++) |
|  | { |
|  | oblast3Forma[i].x += gorod3Pos.x; |
|  | oblast3Forma[i].y += gorod3Pos.y; |
|  | } |
|  | for(int i = 0; i < sizeof(gorod3Forma)/8; i++) |
|  | { |
|  | gorod3Forma[i].x += gorod3Pos.x; |
|  | gorod3Forma[i].y += gorod3Pos.y; |
|  | } |
|  | for(int i = 0; i < sizeof(oblast4Forma)/8; i++) |
|  | { |
|  | oblast4Forma[i].x += gorod4Pos.x; |
|  | oblast4Forma[i].y += gorod4Pos.y; |
|  | } |
|  | for(int i = 0; i < sizeof(gorod4Forma)/8; i++) |
|  | { |
|  | gorod4Forma[i].x += gorod4Pos.x; |
|  | gorod4Forma[i].y += gorod4Pos.y; |
|  | } |
|  |  |
|  | oblast1 = CreatePolygonRgn(oblast1Forma, sizeof(oblast1Forma)/8, WINDING);//создания самих фигур для них |
|  | gorod1 = CreatePolygonRgn(gorod1Forma, sizeof(gorod1Forma)/8, WINDING); |
|  | oblast2 = CreatePolygonRgn(oblast2Forma, sizeof(oblast2Forma)/8, WINDING); |
|  | gorod2 = CreatePolygonRgn(gorod2Forma, sizeof(gorod2Forma)/8, WINDING); |
|  | oblast3 = CreatePolygonRgn(oblast3Forma, sizeof(oblast3Forma)/8, WINDING); |
|  | gorod3 = CreatePolygonRgn(gorod3Forma, sizeof(gorod3Forma)/8, WINDING); |
|  | oblast4 = CreatePolygonRgn(oblast4Forma, sizeof(oblast4Forma)/8, WINDING); |
|  | gorod4 = CreatePolygonRgn(gorod4Forma, sizeof(gorod4Forma)/8, WINDING); |
|  |  |
|  |  |
|  |  |
|  | for(int i = 0; i < passChislo; i++)//создание потоков с человечками |
|  | { |
|  | passajir pas1;//создаем пассажира, настраиваем его параметры |
|  | pas1.fx = rand() % 100 + gorod1Pos.x - 50; |
|  | pas1.fy = rand() % 100 + gorod1Pos.y - 50; |
|  | pas1.x = pas1.fx; |
|  | pas1.y = pas1.fy; |
|  | pas1.sonDlinna = rand()%30000 + 30000; |
|  | passajiry[i] = pas1;//ложим его в масссива |
|  | HANDLE hthr; |
|  | hthr = (HANDLE)\_beginthread(passajirFunction, 0,(void\*)i);//непосредественнно запуск потока и передаем номер ячейки (i) массива с этим пассажиром |
|  | } |
|  |  |
|  | samoletReis.x = gorod4Pos.x; |
|  | samoletReis.y = gorod4Pos.y; |
|  | samoletPolet = TRUE; |
|  | samoletGorodTekush = 3;//создание первоначального самолета с его состоянием |
|  |  |
|  | while(TRUE)//в данном цикле описан алгоритм работы самолета |
|  | { |
|  | if(samoletPolet == TRUE) |
|  | { |
|  | float dx; |
|  | float dy; |
|  | dx = (double)samoletReis.x - samoletPosX; |
|  | dy = (double)samoletReis.y - samoletPosY; |
|  |  |
|  | float l = sqrt(dx\*dx + dy\*dy); |
|  | dx /= l; |
|  | dy /= l;//перемещение по вектору |
|  |  |
|  | if(samoletScorost < l/10) |
|  | { |
|  | samoletScorost++;//ускорение до опр предела |
|  | }else{ |
|  | samoletScorost = l/10; |
|  | } |
|  | if(samoletScorost > 10) |
|  | { |
|  | samoletScorost = 10;//торможение если близко к цели |
|  | } |
|  |  |
|  |  |
|  |  |
|  | samoletPosX += dx\*samoletScorost; |
|  | samoletPosY += dy\*samoletScorost; |
|  | samoletPosRis.x = samoletPosX; |
|  | samoletPosRis.y = samoletPosY; |
|  |  |
|  | if(samoletPosX > samoletReis.x - 1 && samoletPosX < samoletReis.x + 1 && samoletPosY > samoletReis.y - 1 && samoletPosY < samoletReis.y + 1) |
|  | { |
|  | samoletPolet = FALSE;//елси очень очень близко к цели, то самолет на идет на стоянку |
|  | samoletPass = 0; |
|  | samoletSon = 60;//ждет |
|  | samoletStop = TRUE; |
|  | } |
|  | } |
|  |  |
|  | if(samoletSon > 0)//пока самолет на стоянке |
|  | { |
|  | samoletSon--; |
|  | }else{ |
|  | if(samoletGorodTekush == 0 && samoletPolet == FALSE) //елси уже закончилось время ожидания |
|  | { |
|  | samoletGorodTekush = 1;//выбирает следующий город |
|  | samoletReis.x = gorod2Pos.x;//координаты |
|  | samoletReis.y = gorod2Pos.y; |
|  | samoletPolet = TRUE;//и летит |
|  | samoletStop = FALSE; |
|  | } |
|  | if(samoletGorodTekush == 1 && samoletPolet == FALSE) |
|  | { |
|  | samoletGorodTekush = 2; |
|  | samoletReis.x = gorod3Pos.x; |
|  | samoletReis.y = gorod3Pos.y; |
|  | samoletPolet = TRUE; |
|  | samoletStop = FALSE; |
|  | } |
|  | if(samoletGorodTekush == 2 && samoletPolet == FALSE) |
|  | { |
|  | samoletGorodTekush = 3; |
|  | samoletReis.x = gorod4Pos.x; |
|  | samoletReis.y = gorod4Pos.y; |
|  | samoletPolet = TRUE; |
|  | samoletStop = FALSE; |
|  | } |
|  | if(samoletGorodTekush == 3 && samoletPolet == FALSE) |
|  | { |
|  | samoletGorodTekush = 0; |
|  | samoletReis.x = gorod1Pos.x; |
|  | samoletReis.y = gorod1Pos.y; |
|  | samoletPolet = TRUE; |
|  | samoletStop = FALSE; |
|  | } |
|  | } |
|  |  |
|  | if(PeekMessage(&msg, hwnd, 0, 0, PM\_REMOVE))//мониторим события окна |
|  | { |
|  | DispatchMessage(&msg); |
|  | } |
|  |  |
|  | paintFunction();//рисуем все |
|  |  |
|  | Sleep(1000/tScorost);//задержка между кадрами |
|  | } |
|  | } |