****

**КОЛЛЕДЖ МНОГОУРОВНЕВОГО ПРОФЕССИОНАЛЬНОГО**

**ОБРАЗОВАНИЯ**

**Отчет по практической работе**

**По дисциплине:** Основы объектно-ориентированного программирования

**Выполнила:**

Студентка группы 32КС-20

Байкова А.В

**Проверил преподаватель:**

Кукшева Б.А

**Москва 2022 г**

**Содержание**

[**Введение** 3](#_Toc116251448)

[**Практическая работа** 4](#_Toc116251449)

[**Создание графического интерфейса для компьютерных систем** 4](#_Toc116251450)

[**Реализация функционала игры** 5](#_Toc116251451)

[**Заключение** 12](#_Toc116251452)

# **Введение**

Для того чтобы решить поставленную задачу нужно осознавать, о чем идет речь и понимать суть вопроса. Для этого необходимо знать основные понятия о программировании.

Программированием называется процесс создания компьютерных программ. Компьютерная программа является связкой многочисленных строк специального текста. Он является специальным, потому что создан таким образом, чтобы машине было понятно, какие действия должны быть выполнены.

Компьютерный код — это специальный текст, состоящий из набора пошаговых инструкций. Он не всегда содержит в себе нули и единицы, также в нём есть определённые слова и дополнительные символы. Компьютер считывает код, который сообщает ему, какие операции следует выполнить с данными. Для создания программ программист, должен уметь правильно давать команды компьютеру, для этого были придуманы специальные языки- языки программирования.

Язык программирования — формальная знаковая система, предназначенная для записи компьютерных программ. Язык программирования определяет набор лексических, синтаксических и семантических правил, задающих внешний вид программ и действия, которые выполнит исполнитель (компьютер) под её управлением. Получается, что для разных задач нужен определенный язык программирования.

Передо мной стоит задача создания программ для компьютерных систем и интерфейса для удобной работы пользователя с программой. Для выполнения я решил использовать язык С#, так как он прост в освоении и позволяет гибко изменять интерфейс.

## **Практическая работа**

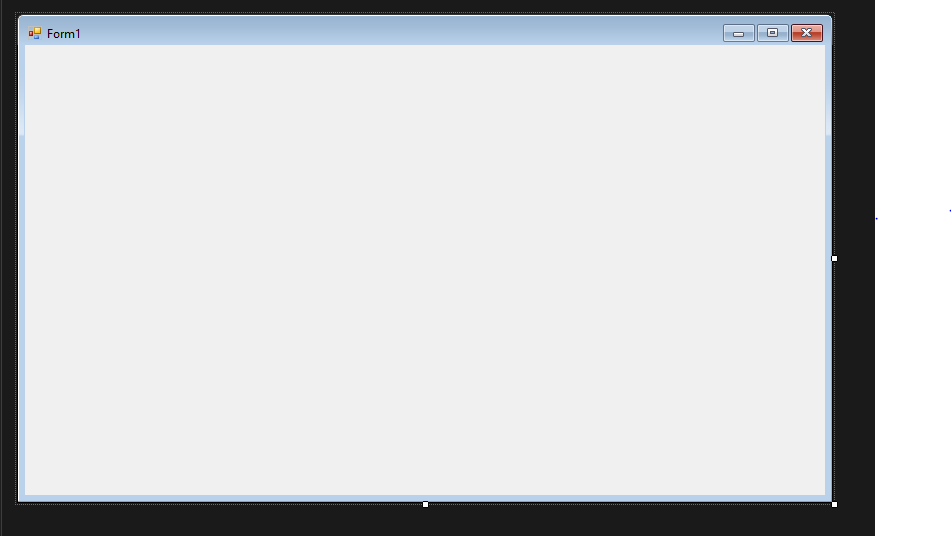
**Цели:** научиться создавать и программировать игру Google Dino.

# **Создание графического интерфейса для компьютерных систем**

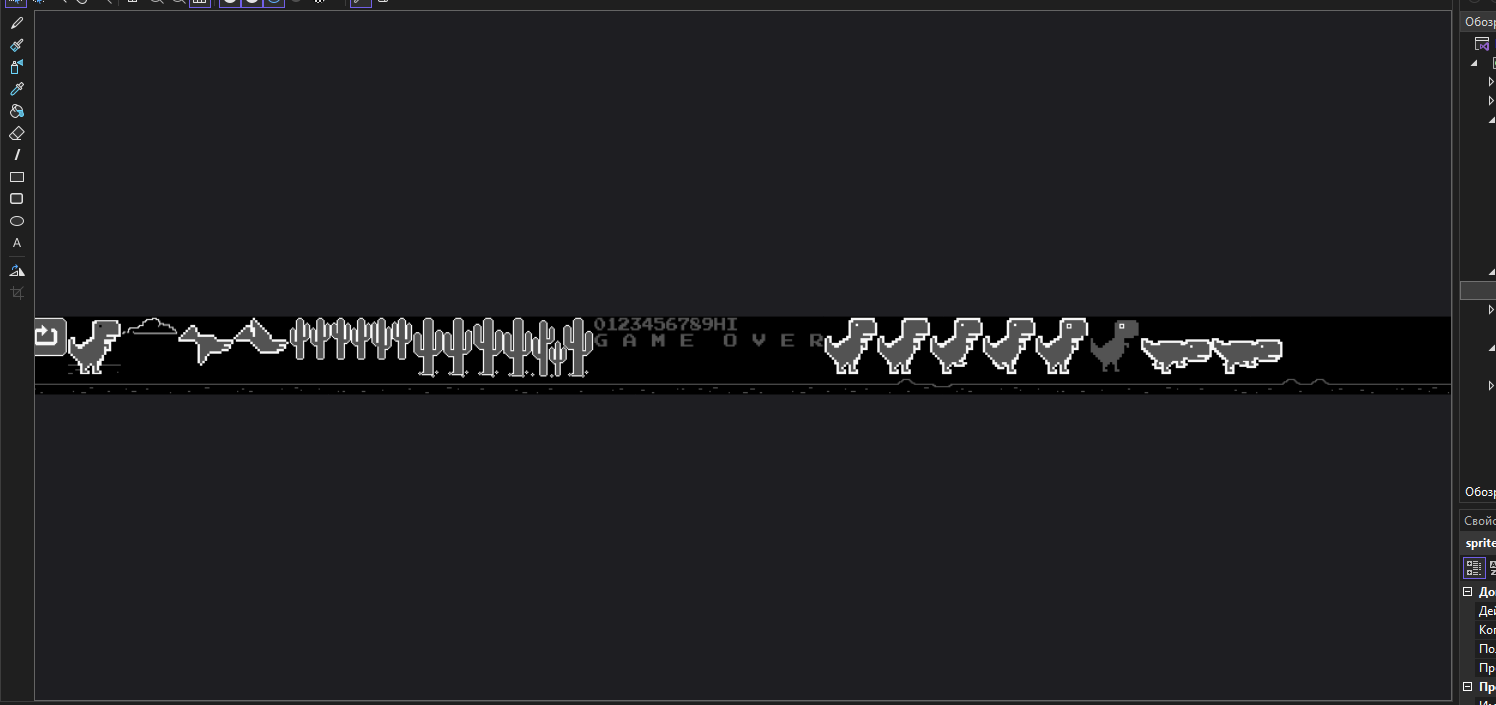
**1**.Создаем основную форму для нашей работы

Вкладка «проект»🡺 добавить форму (Windows Form)

Размеры форм будем изменять под внутренние наполнение элементами.



**2**.Вносим в программу наши изображения персонажей, которые будут присутствовать в будущей игре.



**3.**Начинаем писать программный код для каждого объекта нашей будущей игры

### **Реализация функционала игры**

**1) Bird(птица)**

Отвечает за отображение и реализацию объекта птица

{

public class Bird

{

public Transform transform; **- хранит позицию и размер**

int frameCount = 0;

int animationCount = 0;**- номер анимации которую нужно отобразить**

public Bird(PointF pos,Size size)

{

transform = new Transform(pos, size);

}

public void DrawSprite(Graphics g) **-процесс отвечающий за создание графики**

{

frameCount++;

if (frameCount <= 10)

animationCount = 0; **-будет отображаться первая птица**

else if (frameCount > 10 && frameCount <= 20)

animationCount = 1; **-будет отображаться вторая птица**

else if (frameCount > 20)

frameCount = 0;**- анимация будет обнуляться**

g.DrawImage(GameController.spritesheet, new Rectangle(new Point((int)transform.position.X, (int)transform.position.Y), new Size(transform.size.Width, transform.size.Height)), 264+92\*animationCount, 6, 83, 71, GraphicsUnit.Pixel); );**-указание качества отрисовки**

}

}

}

**2) Cactus(кактус)**

Класс для отображения объекта кактус

{

public Transform transform;

int srcX = 0; **-переменная для выбора рандомного кактуса**

public Cactus(PointF pos,Size size)

{

transform = new Transform(pos, size);

ChooseRandomPic();

}

public void ChooseRandomPic()**-команда которая позволяет сделать выбор**

{

Random r = new Random();

int rnd = r.Next(0, 4);

switch (rnd)

{

case 0:

srcX = 754;

break;

case 1:

srcX = 804;

break;

case 2:

srcX = 704;

break;

case 3:

srcX = 654;

break;

}

}

public void DrawSprite(Graphics g)

{

g.DrawImage(GameController.spritesheet, new Rectangle(new Point((int)transform.position.X, (int)transform.position.Y), new Size(transform.size.Width, transform.size.Height)), srcX, 0, 48, 100, GraphicsUnit.Pixel); );**-указание качества отрисовки**

}

}

**3) Игровой контроллер**

Отвечает за связь объектов в программном коде

{

public static Image spritesheet; **-хранит набор рисунков объектов**

public static List<Road> roads;

public static List<Cactus> cactuses;

public static List<Bird> birds;

public static int dangerSpawn = 10;

public static int countDangerSpawn = 0;

public static void Init() **-создание листов которые будут хранить каждый свое изображение**

{

roads = new List<Road>();

birds = new List<Bird>();

cactuses = new List<Cactus>();

spritesheet = Properties.Resources.sprite;

GenerateRoad();

}

public static void MoveMap()

{

for(int i = 0; i < roads.Count; i++)

{

roads[i].transform.position.X -= 4;

if (roads[i].transform.position.X + roads[i].transform.size.Width < 0)

{

roads.RemoveAt(i);

GetNewRoad();

}

}

for (int i = 0; i < cactuses.Count; i++)

{

cactuses[i].transform.position.X -= 4;

if (cactuses[i].transform.position.X + cactuses[i].transform.size.Width < 0)

{

cactuses.RemoveAt(i);

}

}

for (int i = 0; i < birds.Count; i++)

{

birds[i].transform.position.X -= 4;

if (birds[i].transform.position.X + birds[i].transform.size.Width < 0)

{

birds.RemoveAt(i);

}

}

}

public static void GetNewRoad()

{

Road road = new Road(new PointF(0 + 100 \* 9, 200), new Size(100, 17));

roads.Add(road);

countDangerSpawn++;

if (countDangerSpawn >= dangerSpawn)

{

Random r = new Random();

dangerSpawn = r.Next(5, 9);

countDangerSpawn = 0;

int obj = r.Next(0, 2);

switch (obj)

{

case 0:

Cactus cactus = new Cactus(new PointF(0 + 100 \* 9, 150), new Size(50, 50));

cactuses.Add(cactus);

break;

case 1:

Bird bird = new Bird(new PointF(0 + 100 \* 9, 110), new Size(50, 50));

birds.Add(bird);

break;

}

}

}

public static void GenerateRoad() **-генерация объекта**

{

for(int i = 0; i < 10; i++)

{

Road road = new Road(new PointF(0 + 100 \* i, 200), new Size(100, 17));

roads.Add(road);

countDangerSpawn++;

}

}

public static void DrawObjets(Graphics g)

{

for(int i = 0; i < roads.Count; i++)

{

roads[i].DrawSprite(g);

}

for (int i = 0; i < cactuses.Count; i++)

{

cactuses[i].DrawSprite(g);

}

for (int i = 0; i < birds.Count; i++)

{

birds[i].DrawSprite(g);

}

}

**4) Физика игры**

public class Physics

{

public Transform transform;

float gravity;

float a;

public bool isJumping;

public bool isCrouching;

public Physics(PointF position, Size size)

{

transform = new Transform(position, size);

gravity = 0;

a = 0.4f;

isJumping = false;

isCrouching = false;

}

public void ApplyPhysics()

{

CalculatePhysics();

}

public void CalculatePhysics()

{

if(transform.position.Y<150 || isJumping)

{

transform.position.Y += gravity;

gravity += a;

}

if (transform.position.Y > 150)

isJumping = false;

}

public bool Collide()

{

for(int i = 0; i < GameController.cactuses.Count; i++)

{

var cactus = GameController.cactuses[i];

PointF delta = new PointF();

delta.X = (transform.position.X + transform.size.Width / 2) - (cactus.transform.position.X + cactus.transform.size.Width / 2);

delta.Y = (transform.position.Y + transform.size.Height / 2) - (cactus.transform.position.Y + cactus.transform.size.Height / 2);

if (Math.Abs(delta.X) <= transform.size.Width / 2 + cactus.transform.size.Width / 2)

{

if (Math.Abs(delta.Y) <= transform.size.Height / 2 + cactus.transform.size.Height / 2)

{

return true;

}

}

}

for (int i = 0; i < GameController.birds.Count; i++)

{

var bird = GameController.birds[i];

PointF delta = new PointF();

delta.X = (transform.position.X + transform.size.Width / 2) - (bird.transform.position.X + bird.transform.size.Width / 2);

delta.Y = (transform.position.Y + transform.size.Height / 2) - (bird.transform.position.Y + bird.transform.size.Height / 2);

if (Math.Abs(delta.X) <= transform.size.Width / 2 + bird.transform.size.Width / 2)

{

if (Math.Abs(delta.Y) <= transform.size.Height / 2 + bird.transform.size.Height / 2)

{

return true;

}

}

}

return false;

}

public void AddForce()

{

if (!isJumping)

{

isJumping = true;

gravity = -10;

}

}

}

**5) Dino**

Класс для отображения объекта динозавр и его анимации

public class Player

{

public Physics physics;

public int framesCount = 0;

public int animationCount = 0;

public int score = 0;

public Player(PointF position, Size size)

{

physics = new Physics(position, size);

framesCount = 0;

score = 0;

}

public void DrawSprite(Graphics g)

{

if (physics.isCrouching)

{

DrawNeededSprite(g, 1870, 40, 109, 51, 118, 1.35f);

}

else

{

DrawNeededSprite(g, 1518, 0, 79, 91, 88, 1);

}

}

public void DrawNeededSprite(Graphics g, int srcX, int srcY, int width, int height, int delta, float multiplier)

{

framesCount++;

if (framesCount <= 10)

animationCount = 0;

else if (framesCount > 10 && framesCount <= 20)

animationCount = 1;

else if (framesCount > 20)

framesCount = 0;

g.DrawImage(GameController.spritesheet, new Rectangle(new Point((int)physics.transform.position.X, (int)physics.transform.position.Y), new Size((int)(physics.transform.size.Width \* multiplier), physics.transform.size.Height)), srcX + delta \* animationCount, srcY, width, height, GraphicsUnit.Pixel);

}

}

**6) Road(дорожка)**

Хранит дорожку игры и объекты, которые будут присутствовать на ней.

public class Road

{

public Transform transform;

public Road(PointF pos, Size size)

{

transform = new Transform(pos, size);

}

public void DrawSprite(Graphics g) - **функция для отрисовки дороги**

{

g.DrawImage(GameController.spritesheet, new Rectangle(new Point((int)transform.position.X, (int)transform.position.Y), new Size(transform.size.Width, transform.size.Height)), 2300, 112, 100, 17, GraphicsUnit.Pixel);**-указание качества отрисовки**

}

}

**7)Transform**

Хранит позицию и размер объекта

public class Transform

{

public PointF position;

public Size size;

public Transform(PointF pos,Size size)

{

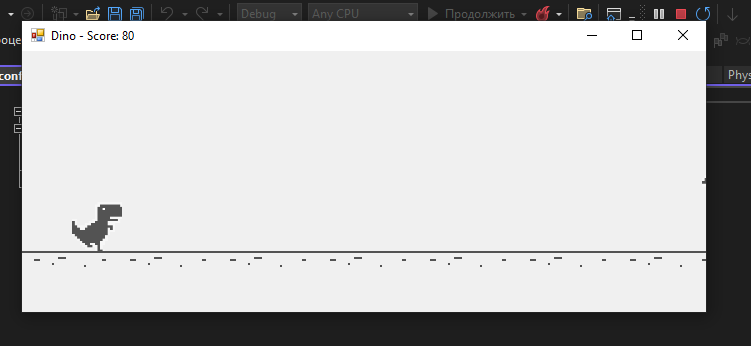
this.position = pos;

this.size = size;

}

}

5)После написанного программного кода уточняем форму запуска, после чего смело нажимаем клавишу запуска и видим созданную нами игру



Программный код данной игры можно получить по ссылке:

**https://github.com/AnastasiyaLan/Dino.git**

## **Заключение**

Передо мной была поставлена задача создания одной из известных игр Google Dino и отработать навыки написания программного кода игры, и как я считаю задача была выполнена в полном объёме. В дальнейшем для улучшения работоспособности данной игры можно будет сделать ее более простой и понятной для любого пользователя. Создать инструкцию с указаниями, стартовое меню и соответствующие дополнительные кнопки управления.