Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

Кафедра ПМ и К

КУРСОВАЯ РАБОТА

По дисциплине «Объектно-ориентированное программирование»

Тема: «Космический 2D шутер в поджанре shoot ‘em up»

Выполнили: студенты группы ИП-912

Лёвкин И. А.

Панин Н. О.

Проверил: доцент кафедры ПМиК

Ситняковская Е. И.

Новосибирск, 2020

Постановка Задачи

Наша цель состояла в разработке космического 2D шутера в поджанре shoot ‘em up с использованием графики (библиотека Monogame), на языке C#, рассчитанную на одного игрока. У нас не было опыта в разработке на C# и Monogame. Но был опыт в разработке графических программ на SFML и C++.

Наш план состоял из:

1. Практичного использования технологий ООП
2. Связывания логики с графическим окружением

Приоритет, естественно, на технологии ООП.

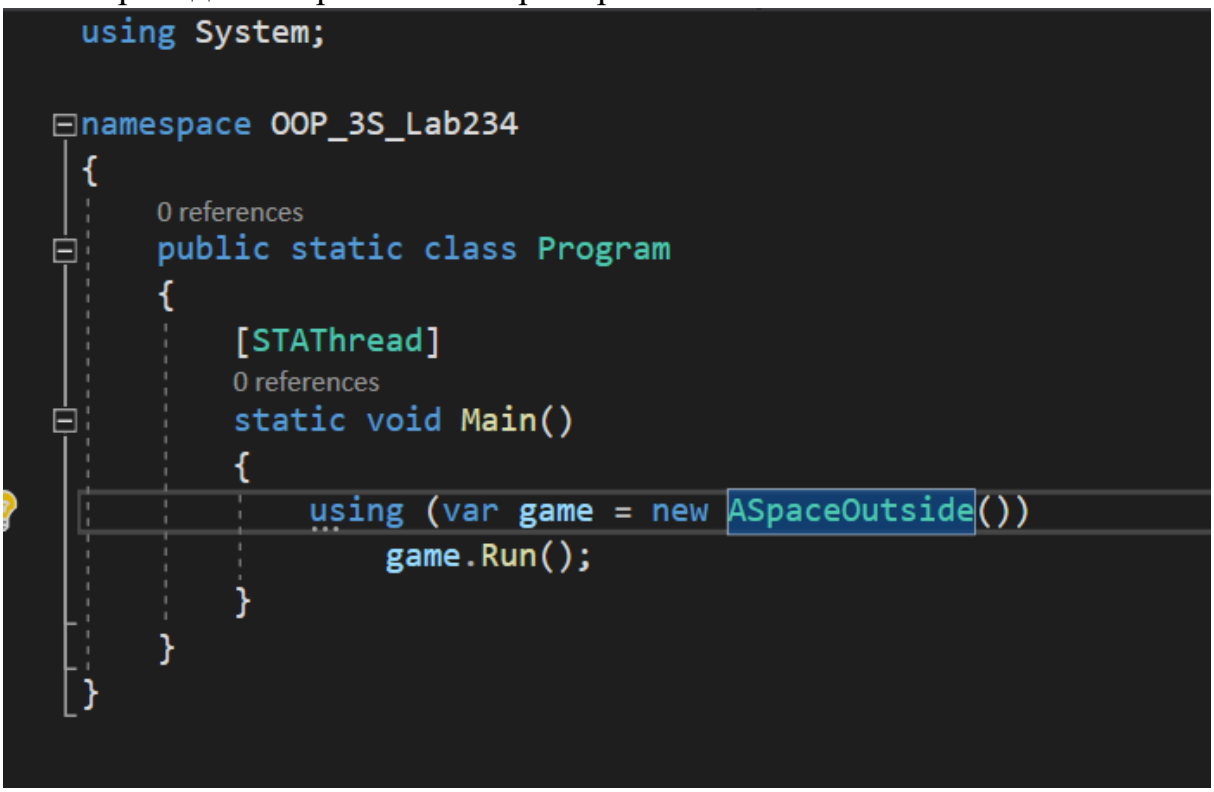
Опыт ООП мы получили благодаря лабораторным работам. А вот опыта с Monogame у нас не было, но в целом, примерное равное количество времени было потрачено на проработку кода классов и отладку графики.

Технологии ООП

У нас в курсовой работе задействованы следующие технологии ООП, а именно:

1. Инкапсуляция
2. Наследование
3. Конструкторы
4. Параметры по умолчанию
5. Абстрактные классы
6. Интерфейсы

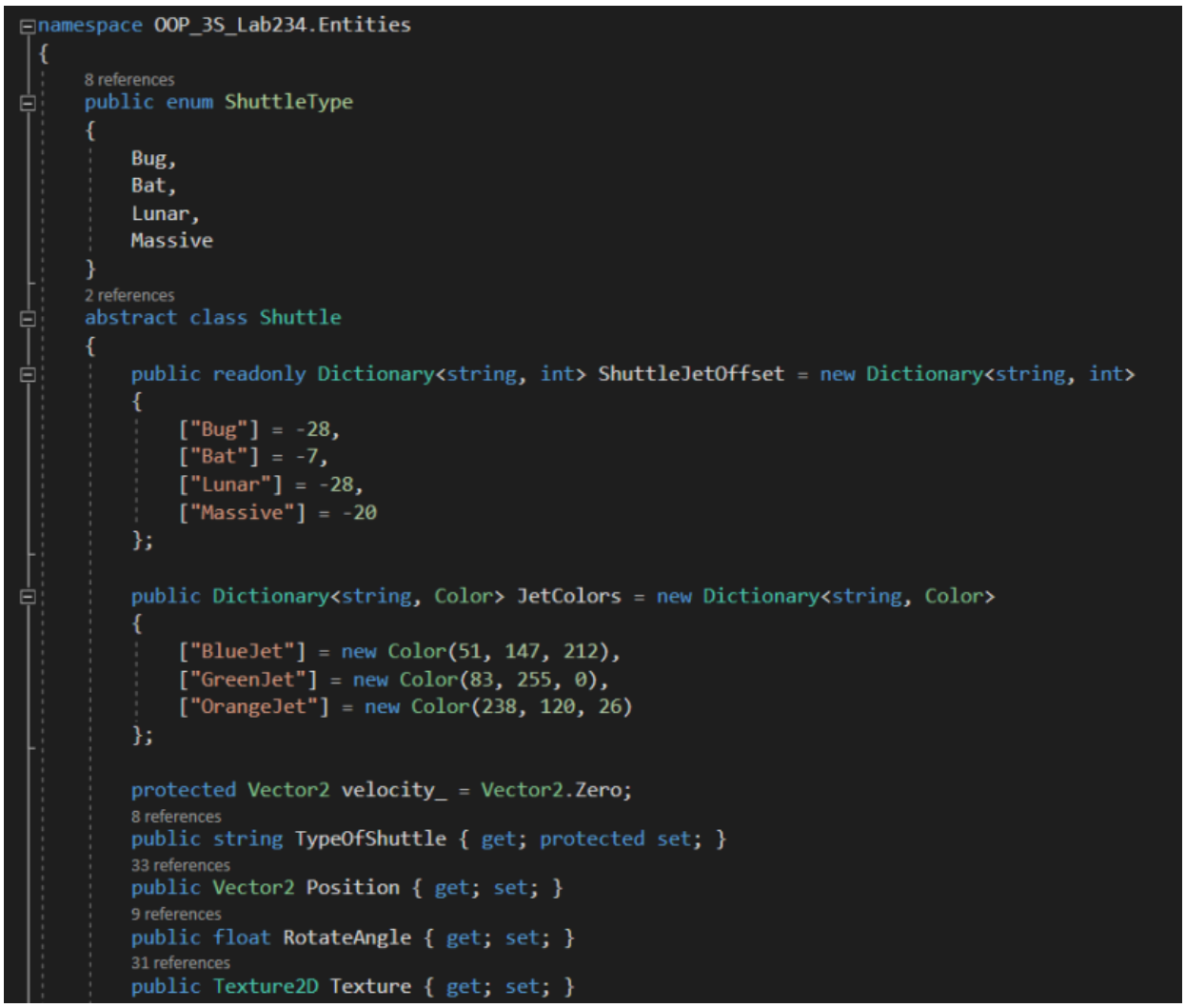
Ниже приведены скриншоты с примерами использования технологий ООП.



```
private SpriteBatch _spriteBatch;
public static Texture2D white;

RenderTarget2D viewport;

Waves waves;

1 reference
public ASpaceOutside()
{
    _graphics = new GraphicsDeviceManager(this);
    Content.RootDirectory = "Content";
    IsMouseVisible = true;
}

0 references
protected override void Initialize()...

0 references
protected override void LoadContent()...

1 reference
protected void LoadGame()...


1 reference
private void GameLoaded(object obj)...

0 references
protected override void Update(GameTime gameTime)
{
    // Switching by State
    switch (State)
    {
        case GameState.LoadingStart:
            break;
        case GameState.MainMenu:
            MainMenu.Update(gameTime, this);
            break;
        case GameState.Gameplay:
            waves.Update(gameTime);
            break;
```

```
GuiText wavesText;
GuiText livesText;
Player player;
List<Enemy> clones;
int wave;
int attempts;
float hitTime;
float waveDelay;

1 reference
public Waves()...

1 reference
public void Init()...

1 reference
public void Load(ContentManager Content, Vector2 resolution)
{
    backgroundTexture = Content.Load<Texture2D>("Images/Backgrounds/background");
    playerShieldTexture = Content.Load<Texture2D>("Images/Particles/shield");

    wavesText = new GuiText(Content.Load<Texture2D>("Images/Controls/guiElementRight"), Content.Load<SpriteFont>("Fonts/default"))
    {
        Position = new Vector2(resolution.X - 256, 0),
        Text = "Wave: 1",
    };

    livesText = new GuiText(Content.Load<Texture2D>("Images/Controls/guiElementLeft"), Content.Load<SpriteFont>("Fonts/default"))
    {
        Position = new Vector2(0, 0),
        Text = "Health: 3",
    };

    gameoverText = new GuiText(Content.Load<Texture2D>("Images/Backgrounds/gameover"), Content.Load<SpriteFont>("Fonts/default"))
    {
        Position = new Vector2(0, 0),
        Text = "",
        PenColour = Color.White
    };

    player.Load(Content);
```

```
namespace OOP_3S_Lab234.Entities
{
    3 references
    class Player : Shuttle
    {
        1 reference
        public Player(Vector2 spawnPoint)
        {
            Jet = new SpeedJet();
            Jet.Speed = 375;
            Position = spawnPoint;
            attackDelay = 0.2f;
            isExist = true;
        }
        1 reference
        public void Load(ContentManager Content)
        {
            Texture = Content.Load<Texture2D>("Images/Shuttle/Body/massiveBody");
            Cabin = Content.Load<Texture2D>("Images/Shuttle/Cabin/brickCabin");
            ColliderTexture = Content.Load<Texture2D>("Images/Backgrounds/white");
            TypeOfShuttle = "Massive";
            Jet.Load(
                ASpaceOutside.jetTexture["SlideBlue"],
                new Dictionary<string, Animation> { ["Working"] = new Animation(Content.Load<Texture2D>("Images/Particles/slideParticles"), 10, 0.1f) }
                );

            projectiles = new List<Projectile>();

            Vector2[] colliderPoints =
            {
                new Vector2(Texture.Bounds.X, Texture.Bounds.Y),
                new Vector2(Texture.Bounds.Width, Texture.Bounds.Y),
                new Vector2(Texture.Bounds.Width, Texture.Bounds.Height),
                new Vector2(Texture.Bounds.X, Texture.Bounds.Height),
            };

            Collider = new PolygonCollider(colliderPoints);
        }
```



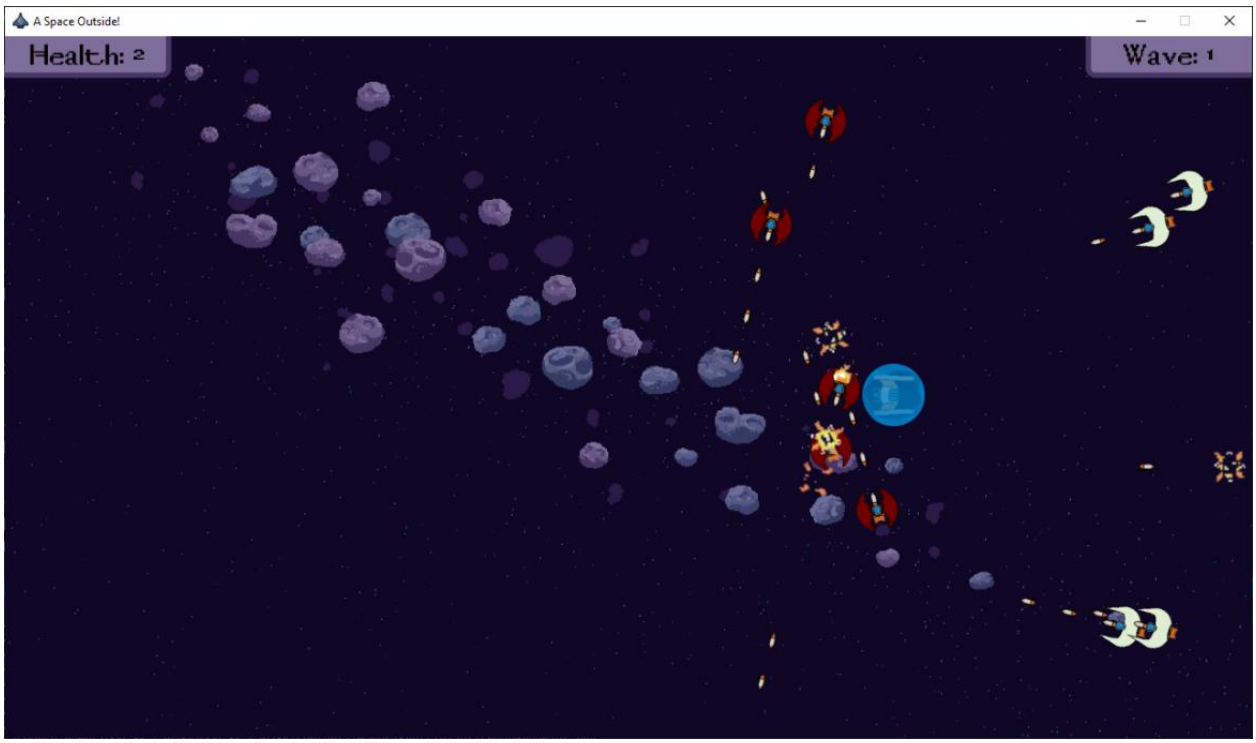
```
namespace OOP_3S_Lab234.Entities
{
    8 references
    public enum ShuttleType
    {
        Bug,
        Bat,
        Lunar,
        Massive
    }
    2 references
    abstract class Shuttle
    {
        public readonly Dictionary<string, int> ShuttleJetOffset = new Dictionary<string, int>
        {
            ["Bug"] = -28,
            ["Bat"] = -7,
            ["Lunar"] = -28,
            ["Massive"] = -20
        };

        public Dictionary<string, Color> JetColors = new Dictionary<string, Color>
        {
            ["BlueJet"] = new Color(51, 147, 212),
            ["GreenJet"] = new Color(83, 255, 0),
            ["OrangeJet"] = new Color(238, 120, 26)
        };

        protected Vector2 velocity_ = Vector2.Zero;
        8 references
        public string TypeOfShuttle { get; protected set; }
        33 references
        public Vector2 Position { get; set; }
        9 references
        public float RotateAngle { get; set; }
        31 references
        public Texture2D Texture { get; set; }
```

```
    31 references
    public Texture2D Texture { get; set; }
    3 references
    public Texture2D Cabin { get; set; }
    7 references
    public bool isDamaged { get; set; }
    2 references
    protected Texture2D ColliderTexture { get; set; }
    protected float attackDelay = 0.2f;
    protected bool isAbleToAttack = true;
    protected float attackTimer = 0;
    protected float projectilesDestroyTimer = 0;
    public List<Projectile> projectiles;
    public PolygonCollider Collider;
    public IJet Jet = new SpeedJet();
    public bool isExist;

    2 references
    public void Draw(SpriteBatch _spriteBatch)...
    2 references
    public virtual void Load(string cabin)...
    3 references
    protected virtual void BorderCollision(Vector2 offset, Vector2 resolution)...

    2 references
    public void Attack()...

    4 references
    public abstract void Update(GameTime gameTime, Vector2 resolution);

    protected float t = 0;
    protected float step = 1;
    protected bool isStoped = false;
    protected bool clearStep = true;
    protected Vector2 tmpVel;
    1 reference
    protected float SmoothStep(float x)
    {
        return (float)Math.Sqrt(1 - Math.Pow(x - 1, 2));
    }
}
```

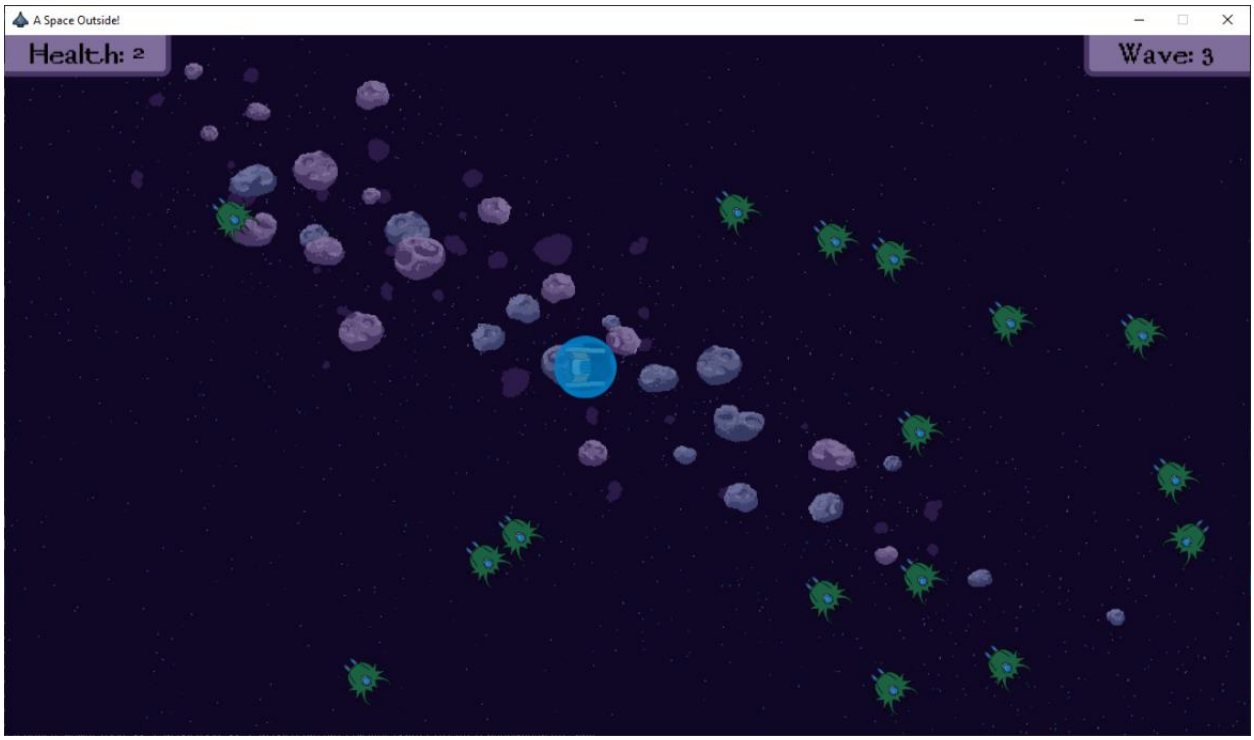
Результат работы

Экран загрузки при открытии в игру

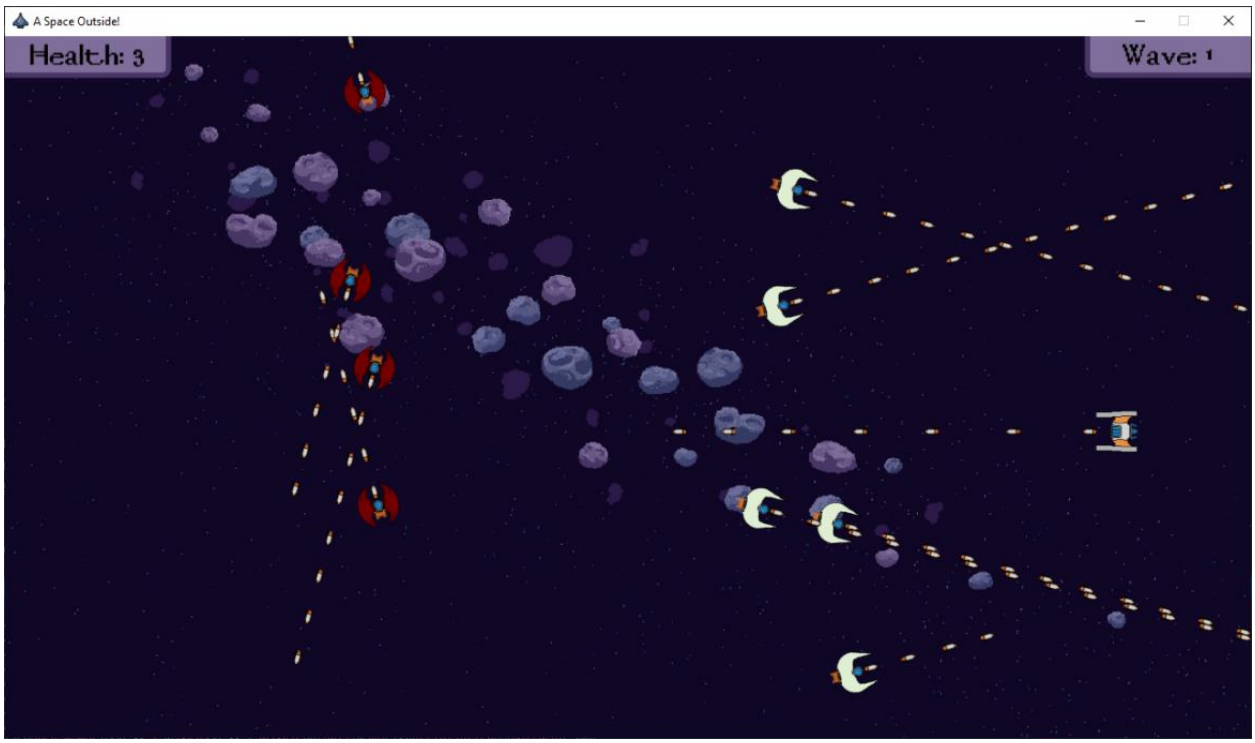
Главное меню игры



Получении урона у игрока на 1 секунду появляется щит



Каждая третья волна – особые противники, в начале волны появляется щит на 2 секунды



Игрок имеет возможность стрелять ракетами так же, как и противники

Выпуская ракеты, игрок должен уничтожать противников для продвижения по волнам.

Заключение

Таким образом, у нас получилось реализовать курсовой проект «Космический 2D шутер в поджанре shoot ‘em up» с использованием технологий ООП, языка C# и библиотеки Monogame. На создание игры у нас ушло около 2-х месяцев с перерывами. Основной проблемой для нас было вычисление столкновений игровых сущностей между собой и их сущностей в виде ракет и т.д.

Несмотря на встретившиеся нам на пути трудности, мы преуспели в выполнении поставленной нами задачи. Были внедрены максимально необходимые технологии ООП, получен опыт работы с языком C# и графической библиотекой Monogame.

Используемые источники

1. https://docs.monogame.net/
2. https://community.monogame.net/
3. https://stackoverflow.com/