Московский Авиационный Институт
(Национальный исследовательский университет)

«Информационные технологии и прикладная математика»

Кафедра вычислительной математики и программирования

Курсовая работа
по курсу «Программирование игр»

Astr Piu Piu (продолжение Astr)

Студент: Лазаревич О.А.

Группа: М8О-108М-20

Преподаватель: Аносова Н.П.

Москва, 2022

Оглавление

Введение

За основу данной работы была выбрана лабораторная работа №3 «Astr». Таким образом помимо уже реализованной части данную работу необходимо доработать по нескольким пунктам: задать цель игры, определить взаимодействие игрока с объектами на сцене.

Основная часть

Данная игра являться бесконечной. При помощи лазерных выстрелов игрок способен уничтожать надвигающиеся астероиды. За каждый уничтоженный астероид игрок получает одно очко. Игрок проигрывает если произошло столкновение с астероидом. Таким образом целью игры является преодоление ранее установленного рекорда по количеству уничтоженных астероидов и одновременно избегание от столкновений с ними.

Текстурирование моделей является важной частью при создании игр. Текстурой называется изображение, накладываемое на определенные части трехмерной модели. Для корабля и астероида создадим модели в Blender. Для материала лазера добавим ещё значение Emission Color которые добавит дополнительное свечение.

В GameObject создадим текстовое поле Text (Unity также создаст объект Canvas если он ранее не был создан без которого невозможно существование пользовательского интерфейса). Расположим текстовое поле в правой нижней части экрана.

Создадим префаб лазерного луча. Лазерный луч представляет собой вытянутый Capsule, который автоматически создаётся при выстреле и летит вперед. При столкновении с астероидом оба уничтожаются. Средствами Unity создадим Capsule. Зададим ему ранее подготовленный материал лазера.

Сцена уже подготовлена, приступим к реализации поведения объектов на ней.

Сперва определим способ обновления счётчика очков. Для этого определим класс `UpdateScore` и зададим его как компонент для текстового поля счётчика очков.

```
public class UpdateScore: MonoBehaviour {
 private static Text _text = null;
```

В методе `Awake` получим и сохраним ссылку на объект представляющий собой текстовое поле пользовательского интерфейса.

```
private void Awake() {
   _text = GetComponent<Text>();
   setScore(0);
 }
```

И определим статический метод для задания текущего счёта.

```
public static void setScore(System.UInt32 score) {
   if(_text != null) {
     _text.text = score.ToString("D8");
   }
 }
```

Каждый раз, когда происходит столкновение с астероидом вызывается этот метод. Параметром передается текущее количество очков. Далее это значение форматируется с добавлением ведущих нулей до общей длины в восемь символов.

Далее определим класс `LaserHit` который будет отвечать за логику лазера в момент столкновения с астероидом и добавим его к префабу лазера.

```
public class LaserHit: MonoBehaviour {
 static private System.UInt16 _score = 0;
```

Данный класс имеет приватное поле `_score` которое хранит значение текущего счёта. Всякий раз, когда лазер сталкивается с астероидом происходит увеличение этого значения на единицу и вызывается статический метод `UpdateScore.setScore`.

```
private void OnCollisionEnter(Collision collision)
 {
   if (collision.gameObject.tag == "Enemy")
   {
     GameObject.Destroy(collision.gameObject);
     GameObject.Destroy(gameObject);
     UpdateScore.setScore(++_score);
   }
 }
```

Кроме того, как уже отмечалось в предыдущей работе каждый астероид имеет тег "Enemy" поэтому обновление счетчика будет происходить только если столкновение произошло с астероидом.

Класс `LaserMovement` будет задавать движение лазерного луча. Этот класс мы также добавим к префабу лазера.

```
public class LaserMovement: MonoBehaviour {
 private Rigidbody _rb = null;
 public System.Single _distance = 1.0F;
 public System.Single _speed = 100.0F;
```

Он функционирует схожим образом с классом, задающим движение астероидов. Однако движение лазерных лучей происходит в противоположном направлении – от летающей тарелки навстречу астероидам).

При достижении заданной границы лазерный луч уничтожается вызовом метода `GameObject.Destroy`.

Осталось лишь задать управление летающей тарелке для выстрелов лазером. Это поведение будет определено внутри класса `UFOFire`.

```
public class UFOFire: MonoBehaviour {
 private System.Single _time = 0.0F;
 public System.Single _fireRate = 1.0F;
```

Данный класс имеет лишь один публичный параметр `_fireRate` который определяет спустя какое время может быть произведен следующий лазерный выстрел.

```
void Update() {
   _time += Time.deltaTime;
   if(_time > _fireRate && Input.GetKeyDown(KeyCode.Space) == true) {
     Vector3 pos = transform.position;
     Instantiate(Resources.Load("LaserCharge"), pos,
Quaternion.identity);
     _time = 0.0F;
   }
 }
```

В методе `Update` происходит проверка времени и проверка на нажатие Пробела. Лазерный луч создаётся посредством метода `Instantiate`. После того как лазерный луч был создан он предоставляется сам себе, а с учётом того что его поведение было заранее описано он знает, что нужно делать…

Заключение

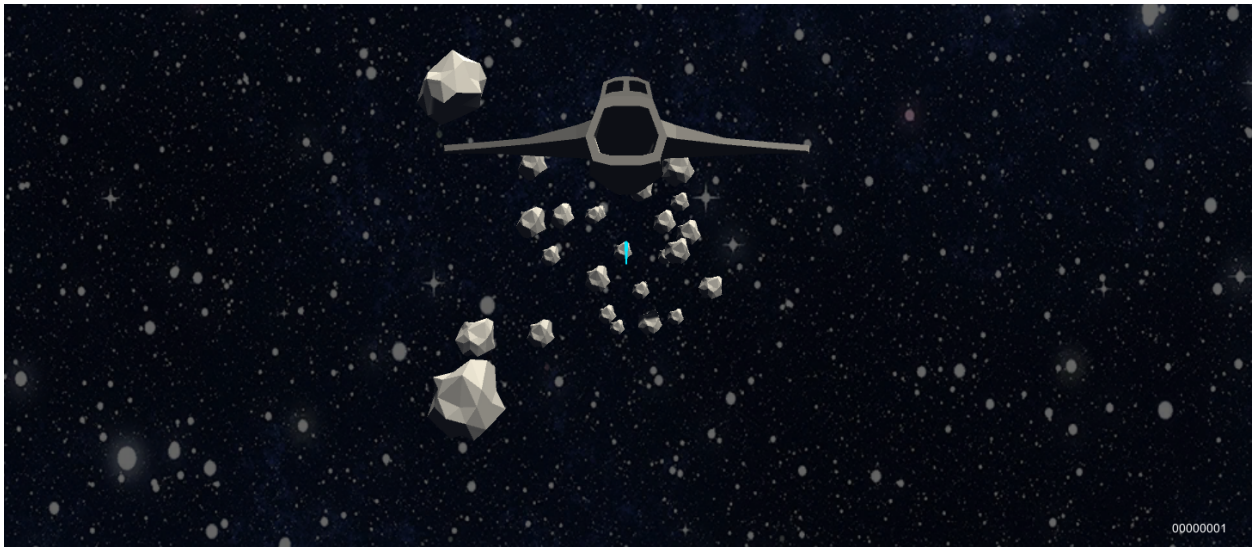
В данной работе были рассмотрены дополнительные этапы при создании игры: с помощью Blender было произведено текстурирование моделей. Был рассмотрен процесс создания материалов в Unity, определена цель игры и задана основная механика.

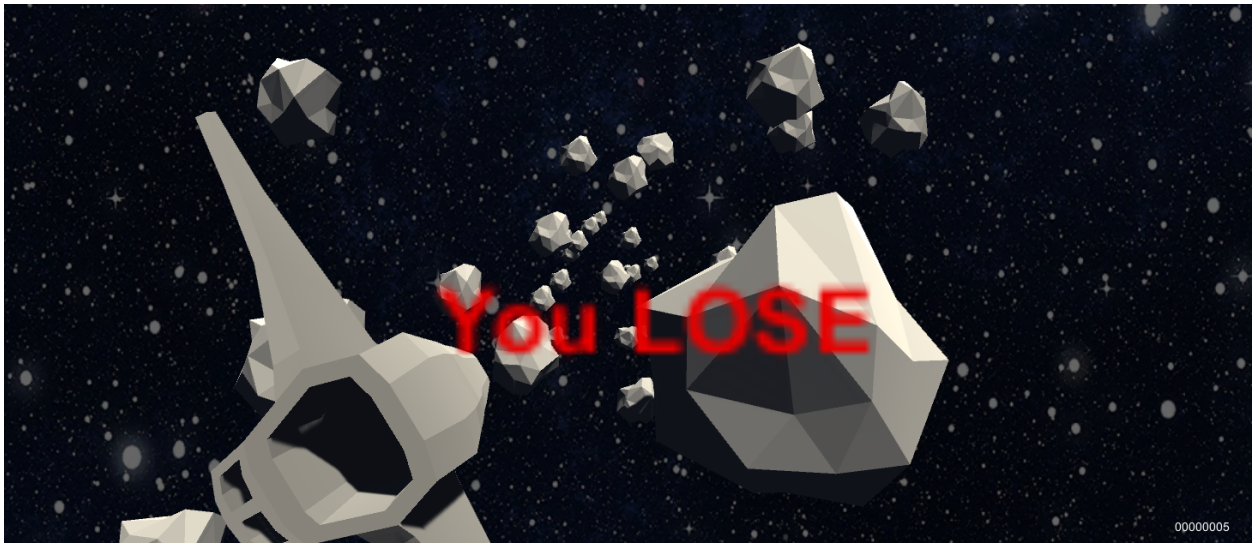
Библиография

1. Unity User Manual — URL: https://docs.unity3d.com/Manual/
2. Unity Tutorials – URL: https://unity3d.com/learn/tutorials
3. Blender Reference Manual – URL: https://docs.blender.org/manual/en/dev/

Приложение

Скриншоты





Листинги

```
// GlobalBehaviour.cs
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GlobalBehaviour: MonoBehaviour {

 void Start()
```

```
 {
   StartCoroutine(create());
 }


 IEnumerator create()
 {
   while (true)
   {
     Vector3 pos = new Vector3(
       Random.Range(-_width, _width),
       Random.Range(-_height, _height),
       _distance);

     GameObject obj =
       (GameObject)Instantiate(Resources.Load("Rock"), pos,
Quaternion.identity);

     RockBehaviour rock = obj.GetComponent<RockBehaviour>();
     rock._speed = Random.Range(_speedMin, _speedMax);
     rock._rotationSpeed = Random.Range(_rotationSpeedMin,
_rotationSpeedMax);

    yield return new WaitForSeconds(0.15f);
   }
 }

 public System.Single _respawnTime = 1.0F;
 public System.Single _distance = 100.0F;
 public System.Single _speedMin = 10.0F;
 public System.Single _speedMax = 30.0F;
 public System.Single _rotationSpeedMin = 1.0F;
 public System.Single _rotationSpeedMax = 10.0F;
 public System.Single _width = 10.0F;
 public System.Single _height = 10.0F;

}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

// LaserHit.cs
public class LaserHit: MonoBehaviour {
```

```
 static private System.UInt16 _score = 0;

 void Awake() {
 }

 private void OnCollisionEnter(Collision collision)
 {
   if (collision.gameObject.tag == "Enemy")
   {
     GameObject.Destroy(collision.gameObject);
     GameObject.Destroy(gameObject);
     UpdateScore.setScore(++_score);
   }
 }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

// LaserMovement.cs
public class LaserMovement: MonoBehaviour {
 private Rigidbody _rb = null;
 public System.Single _distance = 1.0F;
 public System.Single _speed = 100.0F;

 void Awake() {
   _rb = GetComponent<Rigidbody>();
 }

 void Update() {
   if(_rb != null) {
     Vector3 pos = _rb.position;
     pos.z += _speed * Time.deltaTime;
     if(pos.z > _distance) {
       GameObject.Destroy(gameObject);
       return;
     }
     _rb.MovePosition(pos);
   }
 }
}

using System.Collections;
```

```
using System.Collections.Generic;
using UnityEngine;

// RockBehaviour.cs
public class RockBehaviour: MonoBehaviour {

 void Awake() {
   _rigidbody = GetComponent<Rigidbody>();

   _eulerAngles = new Vector3(
     Random.Range(-1.0F, 1.0F),
     Random.Range(-1.0F, 1.0F),
     Random.Range(-1.0F, 1.0F)
   );
 }

 void FixedUpdate () {

   Vector3 pos = _rigidbody.position;
   pos.z -= _speed * Time.deltaTime;
   _rigidbody.MovePosition(pos);

   Vector3 rot = _rigidbody.rotation.eulerAngles;
   rot += _eulerAngles * _rotationSpeed * Time.deltaTime;
   _rigidbody.MoveRotation(Quaternion.Euler(rot));

   if(pos.z < _deadEnd)
     Object.Destroy(gameObject);
 }

 private Rigidbody _rigidbody;

 public System.Single _speed = 1.0F;
 public System.Single _rotationSpeed = 1.0F;
 public System.Single _deadEnd = -10.0F;

 private Vector3 _eulerAngles;

}

// UFOFire.cs
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
```

```
public class UFOFire: MonoBehaviour {
 private System.Single _time = 0.0F;
 public System.Single _fireRate = 1.0F;

 void Update() {
   _time += Time.deltaTime;
   if(_time > _fireRate && Input.GetKeyDown(KeyCode.Space) == true) {
     Vector3 pos = transform.position;
     Instantiate(Resources.Load("LaserCharge"), pos,
Quaternion.identity);
     _time = 0.0F;
   }
 }
}

// UFOInputBehaviour.cs
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class UFOInputBehaviour: MonoBehaviour {

 void Awake() {
   _rb = GetComponent<Rigidbody>();
   _sadText.SetActive(false);
 }
  void FixedUpdate() {
   System.Single inpX = Input.GetAxis("Horizontal");
   System.Single inpY = Input.GetAxis("Vertical");

   Vector3 pos = _rb.position;

   pos.x += inpX * _speed * Time.deltaTime;
   pos.y += inpY * _speed * Time.deltaTime;

   pos.x = Mathf.Clamp(pos.x, -_amplitude, _amplitude);
   pos.y = Mathf.Clamp(pos.y, -_amplitude*0.6F, _amplitude*0.4F);

   _rb.MovePosition(pos);
 }

 private void OnCollisionEnter(Collision collision)
 {
```

```
    if (collision.gameObject.tag == "Enemy")
    {
      _sadText.SetActive(true);
    }
  }

  public System.Single _speed = 1.0F;
  public System.Single _amplitude = 1.0F;
  private Rigidbody _rb;
  public GameObject _sadText;
}

// UpdateScore.cs
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class UpdateScore: MonoBehaviour {
  private static Text _text = null;
  private void Awake() {
    _text = GetComponent<Text>();
    setScore(0);
  }

  public static void setScore(System.UInt32 score) {
    if(_text != null) {
      _text.text = score.ToString("D8");
    }
  }
}
```