Инженерная школа информационных технологий и робототехники

Направление 09.03.01 «Информатика и вычислительная техника»

Отделение информационных технологий

Отчёт по теме

**«Разработка игровых приложений на Unity»**

по дисциплине «Творческий проект»

Выполнил:

студент гр. 8В11 \_\_\_\_\_\_\_\_\_\_\_\_\_ Топчеева Д.А.

\_\_\_.\_\_\_.\_\_\_\_\_\_

Проверил:

Ассистент отделения ИТ

оценка (до 30 баллов): \_\_\_\_\_\_\_\_\_\_\_\_\_

к защите допускаю \_\_\_\_\_\_\_\_\_\_\_\_\_ Коровкин В. А.

\_\_\_.\_\_\_.\_\_\_\_\_\_

**Содержание**

[Введение 3](#_Toc124789475)

[Цели и задачи 3](#_Toc124789476)

[1. Свет в Unity 4](#_Toc124789477)

[1.1. 2D Освещение 4](#_Toc124789478)

[1.2. 3D Освещение 5](#_Toc124789479)

[1.3. Карты нормалей 6](#_Toc124789480)

[2. Разработка игрового приложения 9](#_Toc124789481)

[2.1. Смена дня и ночи 9](#_Toc124789482)

[2.2. Свет и тень от других объектов 10](#_Toc124789483)

[Заключение 13](#_Toc124789484)

[Список источников: 14](#_Toc124789485)

[Приложения: 16](#_Toc124789486)

# Введение

Свет и тень играют важную роль в восприятии мира человеком. Ясный день дает большее ощущение безопасности нежели темная ночь. Эта разность в восприятии окружающей среды часто используется в игровых приложениях.

Световой дизайн применим к играм любого жанра: стратегии, платформеры, головоломки, хорроры, ролевые игры с открытым миром.  Свет позволяет помечать опасности, управлять эмоциями игрока, создавать зоны внимания на уровне, может создавать атмосферу и выступать как часть истории мира.

В данном семестре планируется разработать свет и тени для игрового приложение на Unity.

# Цели и задачи

Добавление света и тени в игровое приложения на Unity

Для выполнения данной цели были поставлены следующие задачи:

1. Изучить среду разработки Unity.
2. Разработка игрового приложения.
   1. Смена дня и ночи.
   2. Свет и тень от других объектов.
3. Оптимизация и исправление ошибок.

# Свет в Unity

# 2D Освещение

При использовании [Universal Render Pipeline](https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@latest/index.html) (URP) с выбранным средством **2D Renderer компонент Light 2D** представляет способ применения 2D-оптимизированного освещения к спрайтам.

Вы можете выбрать один из нескольких различных типов освещения с помощью компонента **Light 2D.**В настоящее время в пакете доступны следующие типы освещения:

* [Freeform](https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@7.1/manual/LightTypes.html#freeform). Вы можете редактировать форму этого типа Light.
* [Sprite](https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@7.1/manual/LightTypes.html#sprite). Вы можете выбрать Sprite для создания этого типа света.
* [Parametric](https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@7.1/manual/LightTypes.html#parametric). Вы можете использовать n-сторонний многоугольник для создания этого типа 2D-освещения.
* [Point](https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@7.1/manual/LightTypes.html#point). Вы можете контролировать внутренний и внешний радиус, направление и угол этого типа света.
* [Global](https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@7.1/manual/LightTypes.html#global). Этот 2D-свет влияет на все визуализированные спрайты на всех целевых слоях сортировки.

Освещение будет только для объектов чей слой находится в списке самого источника света.

Проблема в использовании данного типа освещения в том, что приложение не полностью в 2D (для создания перспективы объекты находятся под углом относительно плоскости мира), из-за чего тени и свет отображаются некорректно или вовсе исчезают, поэтому было решено не использовать этот тип.



Рисунок 1. Объекты находятся под углом к плоскости.

# 3D Освещение

В Unity3D доступно 5 типов источников света:

* Directional Light. Самый простой, имитирует солнечный свет. Представляет из себя бесконечное множество параллельных друг другу лучей.
* Point Light. Точечный источник света, то есть лучи расходятся во все стороны из одной точки. Хорошим примером такого источника света будет обычная лампочка.
* Area Light. Источник света, имеющий площадь. Представьте себе прямоугольную панель, из которой исходит свет, это и будет area light. Такие источники света чаще всего используются в офисах, торговых центрах и других нежилых помещениях, где надо освещать большие пространства.
* Ambient Light. Заполняющий свет, не имеющий источника. Примеры использования: осветить слишком тёмные тени; добавить атмосферности подземелью, заполнив его едва заметным светом биолюминисцентных растений.
* Light Probes. Особый источник света, влияющий исключительно на динамические объекты.

В Unity все объекты делятся на динамические (dynamic) и статические (static). Статическими объектами называются те, которые всегда стоят на месте и никуда не смещаются. Именно для них происходит “запечение” освещения. Динамические объекты — это те, которые наоборот находятся в движении. Сюда входят такие элементы сцены, как герой, монстр, колышущийся флаг, падающий с обрыва камень.

Был выбран данный тип освещения, т.к. в отличие от предыдущего типа свет работал корректно для всех объектов. Но тени при данном типе освещения строятся только от трехмерных моделей, а не от спрайтов, поэтому тени частично были написаны самостоятельно.

# Карты нормалей

Normal mapping — техника, позволяющая изменять нормаль отображаемого пикселя основываясь на цветной карте нормалей, в которой эти отклонения хранятся в виде [текселя](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D0%B5%D0%BB_(%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%BA%D0%B0)" \o "Тексел (графика)), цветовые составляющие которого [r,g,b] интерпретируются в оси вектора [x, y, z], на основе которого вычисляется нормаль, используемая для расчёта освещённости пикселя.

Если разделить все каналы получим видимости направленного освещения:

* Красный канал (Red) – освещение точно справа.
* Зеленый канал (Green) – освещение точно сверху.
* Синий канал (Blue) – освещение точно cпереди.

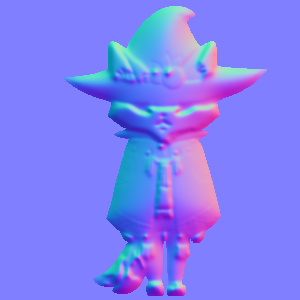


Рисунок 2. Карта нормалей.

**Плюсы:**

* Позволяют создавать cглаженные углы.
* В сочетании с затемнениями на текстуре цвета отлично передают вдавленности.
* Хорошо передают мелкие детали поверхности.
* Хорошо передают узкие и глубокие швы между деталями.
* Позволяет создать иллюзию объема на двумерном изображении.

**Минусы:**

* Не изменяют силуэт.
* Плохо сглаживают геометрию.

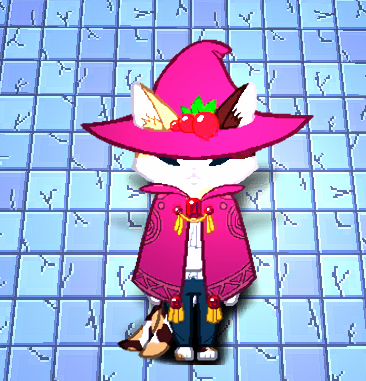


Рисунок 3. Без карты нормалей.



Рисунок 4. С картой нормалей.

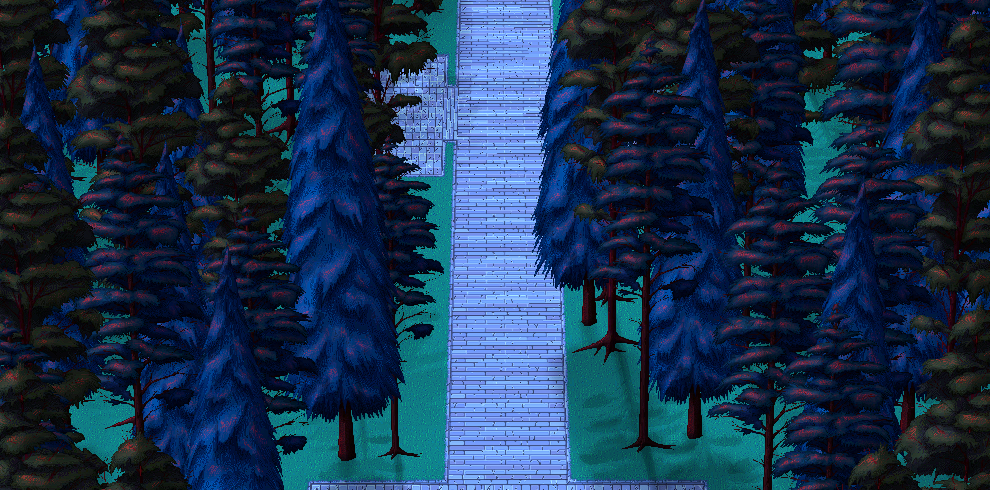
# Разработка игрового приложения

# Смена дня и ночи

Во время разработки была реализована механика смены дня и ночи и движения тени в зависимости от времени.

Сама смена времени суток осуществляется посредством изменения интенсивности источника света, который был выбран как «солнце» (был выбран Directional Light.). Так ночью интенсивность будет равна 0, а днем 1.

Также в зависимости от времени будет изменятся и цвет света, так утром и вечером он будет более теплым чем днем (Приложение 2).



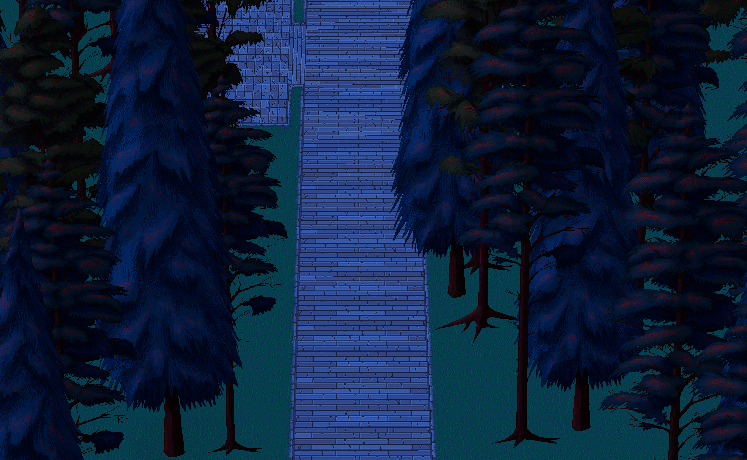


Рисунок 5 и 6. День и ночь.

Движение теней осуществляется посредством изменения их угла по x и y координатам относительно их точки вращения. Также в зависимости от времени будет изменяться и яркость теней (Приложение 3).

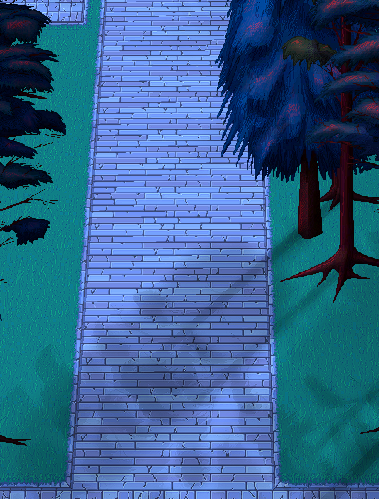


Рисунок 7 и 8. Положение тени относительно времени суток (рис. 7 18:00, рис. 8 9:00)

# Свет и тень от других объектов

Помимо солнца объекты отбрасывают тени также и от других источников света. Для этого была написана программа, которая ищет ближайшие 3 источника света, чьи коллайдеры входят в радиус видимости объекта, и определяет направление тени относительно них (Приложение 4).

Также программа изменяет длину тени, так как источники света могут находиться выше чем объект.

В области видимости может находиться несколько источников света, поэтому яркость теней будет изменяться в зависимости количества источников и времени суток.





Рисунок 9 и 10. Свет от других объектов (рис. 9 один источник света, рис. 10 несколько источников света)



Рисунок 11. Тени днем

Проблема возникла в том, что была слишком большая нагрузка из-за такого количества теней. Решением проблемы было скрытие теней, если нет никаких источников света, или, если есть только один или два, то ненужные тени скрывались.

Также в ходе работы возникла проблема с программой, написанной в прошлом семестре. Для того чтобы освещение работало, к каждому объекту был добавлен материал с картой нормалей. Из-за этого объекты, которые должны были становиться прозрачными, этого не делали. Чтобы решить проблему, надо было брать альфа-канал материала, а не спрайта, который лежал на объекте (Приложение 1).

# Заключение

За текущий семестр были изучены основные моменты работы со светом в среде разработки Unity. Были изучены материалы для работы с объектами на данной платформе. Были выполнены задачи, поставленные на этот семестр. Возникшее в процессе проблемы были решены.

На следующий семестр планируются такие задачи:

1. Добавление механик взаимодействия персонажа с миром.
2. Добавление новых элементов окружающей среды.
3. Добавление анимации.

Подведя итог, можно сказать, что в следующем семестре планируется больше разработать механику взаимодействия с миром и реализовать front-end составляющую приложения. Таким образом будет продолжена работа над данным проектом в будущем.

# Список источников:

* 1. Тьма и свет: световой дизайн в видеоиграх// dtf.ru - Samael Ophidian [Электронный ресурс] -Режим доступа: <https://dtf.ru/gamedev/759245-tma-i-svet-svetovoy-dizayn-v-videoigrah> (Дата обращения 14.01.2023)
  2. Под капотом Graveyard Keeper: Как реализованы графические эффекты // vk.com [Электронный ресурс] -Режим доступа: [https://vk.com/ @scienceandlive-pod-kapotom-graveyard-keeper-kak-realizovany-graficheskie-ef](https://vk.com/@scienceandlive-pod-kapotom-graveyard-keeper-kak-realizovany-graficheskie-ef) (Дата обращения 05.10.2022)
  3. Unity - Как найти ближайший объект // youtube.com - Андрей [Электронный ресурс] -Режим доступа: [https://www.youtube.com/watch?v= qNoOrDb\_peY](https://www.youtube.com/watch?v=%20qNoOrDb_peY) (Дата обращения 25.11.2022)
  4. Вращение объекта вокруг другого объекта//cyberforum.ru [Электронный ресурс] -Режим доступа: <https://www.cyberforum.ru/unity/thread2206350.html> (Дата обращения 20.10.2022)
  5. OverlapShpere - УРОКИ ПО ЮНИТИ / LESSONS ON UNITY // youtube.com [Электронный ресурс] -Режим доступа: [https://www.youtube.com/watch?v= MyMbd\_Hk\_fk](https://www.youtube.com/watch?v=%20MyMbd_Hk_fk) (Дата обращения 14.12.2022)
  6. Удалить / скрыть объект // know-online.com [Электронный ресурс] -Режим доступа: <https://know-online.com/post/unity-ubrat-udalit-object> (Дата обращения 20.06.2022)
  7. Обзор освещения // unity.com [Электронный ресурс] -Режим доступа: <https://docs.unity3d.com/ru/530/Manual/Lighting.html> (Дата обращения 05.10.2022)
  8. Форум Unity// unity.com [Электронный ресурс] -Режим доступа: <https://forum.unity.com/> (Дата обращения 10.01.2023)
  9. Гайд по оптимизации мобильных игр в Unity // dtf.ru - Илья Сергеич [Электронный ресурс] -Режим доступа: <https://dtf.ru/s/unity/293563-gayd-po-optimizacii-mobilnyh-igr-v-unity> (Дата обращения 10.01.2023)
  10. Cистема вопросов и ответов о программировании // stackoverflow.com [Электронный ресурс] -Режим доступа: <https://ru.stackoverflow.com/> (Дата обращения 10.01.2023)
  11. Изменить размер объекта динамически // cyberforum.ru [Электронный ресурс] -Режим доступа: <https://www.cyberforum.ru/unity/thread1591758.html> (Дата обращения 13.12.2022)
  12. Особенности применения карт нормалей // render.ru [Электронный ресурс] -Режим доступа: <https://render.ru/ru/articles/post/10701> (Дата обращения 13.01.2023)
  13. Introduction to Lights 2D // unity.com [Электронный ресурс] -Режим доступа: [https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@7.1/ manual/Lights-2D-intro.html](https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@7.1/%20manual/Lights-2D-intro.html) (Дата обращения 15.12.2023)

# Приложения:

1. Измененная программа, отвечающая за прозрачность объектов:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class ObjTransparency : MonoBehaviour

{

public SpriteRenderer \_sprite;

bool ObjectsOnTrigger;

public float speed;

void FixedUpdate()

{

var color = \_sprite.color;

var mat = GetComponent<SpriteRenderer>().material;

var Mat = mat.color;

if (ObjectsOnTrigger)

{

color.a -= speed \* Time.deltaTime;

Mat.a -= speed \* Time.deltaTime;

}

else

{

color.a += speed \* Time.deltaTime;

Mat.a += speed \* Time.deltaTime;

}

color.a = Mathf.Clamp(color.a, 0.2f, 1);

Mat.a = Mathf.Clamp(Mat.a, 0.2f, 1);

\_sprite.color = color;

mat.color = color;

}

void OnTriggerStay2D(Collider2D other)

{

ObjectsOnTrigger = true;

}

void OnTriggerExit2D(Collider2D other)

{

ObjectsOnTrigger = false;

}

}

1. Программа, отвечающая за солнце:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Sun : MonoBehaviour

{

public float time, speed,intensity;

public Light sunlight;

public Color color1, color2;

Color rescolor;

// Start is called before the first frame update

void Start()

{

sunlight = GetComponent<Light>();

intensity = sunlight.intensity;

}

// Update is called once per frame

void Update()

{

if ((time <= 4) || (time >= 22.4f)) //Ночное время

{

sunlight.intensity = 0;

rescolor = color1;

}

else if ((time >= 8.4f) && (time <= 18)) //День

{

sunlight.intensity = 1;

rescolor = color2;

}

else if ((time > 4) && (time < 8.4f)) //Рассвет

{

sunlight.intensity = (time - 4) / 4.4f;

rescolor = new Color((color2.r - color1.r) \* (time - 4) / 4.4f + color1.r, (color2.g - color1.g) \* (time - 4) / 4.4f + color1.g, (color2.b - color1.b) \* (time - 4) / 4.4f + color1.b);

}

else if ((time > 18) && (time < 22.4)) //Закат

{

sunlight.intensity = (22.4f - time) / 4.4f;

rescolor = new Color((color1.r - color2.r) \* (time-18) / 4.4f + color2.r, (color1.g - color2.g) \* (time-18) / 4.4f + color2.g, (color1.b - color2.b) \* (time-18) / 4.4f + color2.b);

}

intensity = Mathf.Clamp(intensity, 0, 1);

if (time <= 24) //Счетчик времени

time += speed \* Time.deltaTime;

else

time = 0;

sunlight.color = rescolor; //Смена цвета

}

}

1. Программа, отвечающая за движение теней от солнца:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class shadowmoving : MonoBehaviour

{

public GameObject go;

public SpriteRenderer \_sprite;

private Vector3 scale; // Переменная для текущего размера

private Vector3 newscale;

private Sun timesun;

public float time, maxa,speed,angle;

float intensity,y,maxy,x,minx;

// Start is called before the first frame update

void Start()

{

scale = new Vector3(go.transform.localScale.x, go.transform.localScale.y, go.transform.localScale.z); // Проверяем текущий размер

newscale = scale;

GameObject timeday = GameObject.Find("Sun");

timesun = timeday.GetComponent<Sun>();

speed = timesun.speed;

y = scale.y;

maxy = y \* 2;

x = scale.x;

minx = x / 2;

}

// Update is called once per frame

void Update()

{

time = timesun.time;

if (time <= 24) //Счетчик времени

time += speed \* Time.deltaTime;

else

time = 0;

if ((time <= 4) || (time >= 22.4f)) //Ночное время

{

intensity = 0;

}

else if ((time >= 8.4f) && (time <= 18)) //День

{

intensity = 1;

}

else if ((time > 4) && (time < 8.4f)) //Рассвет

{

intensity = (time - 4) / 4.4f;

}

else if ((time > 18) && (time < 22.4)) //Закат

{

intensity = (22.4f - time) / 4.4f;

}

intensity = Mathf.Clamp(intensity, 0, 1);

var color = \_sprite.color;

color.a = intensity \* maxa;

color.a = Mathf.Clamp(color.a, 0, maxa);

\_sprite.color = color;

}

void FixedUpdate()

{

if (((time <= 1.2f)&&(time>=0))||(time>22.4f))

{

float timer=time;

if (time>22.4f)

{

timer = time-22.4f;

}

else if (time<=1.2f)

{

timer = 1.6f + time;

}

newscale = new Vector3(x - minx\* (2.8f - timer) / 2.8f, y+y\*(2.8f-timer)/2.8f, 0);

angle = 0+90\*(2.8f-timer)/2.8f;

}

else if ((time <= 22.4f)&&(time>13.2f))

{

newscale = new Vector3(x - minx \* (time - 13.2f) / 9.2f, y+y\*(time-13.2f)/9.2f, 0f);

angle = 180-90\* (time - 13.2f) / 9.2f;

}

else if ((time <= 13.2f)&& (time > 4))

{

newscale = new Vector3(minx + minx \* (time - 4) / 9.2f, maxy-y\*(time-4)/9.2f, 0f);

angle = 270 - 90 \* (time - 4) / 9.2f;

}

else if ((time <= 4) && (time > 1.2f))

{

newscale = new Vector3(x - minx \* (time - 1.2f) / 2.8f, y+y\*(time-1.2f)/2.8f, 0f);

angle = 360 - 90 \* (time - 1.2f) / 2.8f;

}

go.transform.localScale = newscale;

go.transform.localEulerAngles = new Vector3(20, 0, angle);

}

}

1. Программа, отвечающая за тени от других источников света:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class lightshadow : MonoBehaviour

{

Collider closest, cl1, cl2;

public LayerMask itemsLayer;

bool LightsOnTrigger;

public SpriteRenderer \_sprite, \_sprite2, \_sprite3;

private Vector3 scale; // Переменная для текущего размера

private Vector3 newscale;

private Vector3 position, diff;

private float distance, k, scaleparent, scalelight, y, intensity;

private Light sunlight;

public float angle, radius, ang, h, l;

void Start()

{

scaleparent = transform.localScale.z \* l \* Mathf.Sin(20 \* Mathf.PI / 180); // Проверяем текущий размер самого объекта

y = transform.localScale.y;

GameObject timeday = GameObject.Find("Sun");

sunlight = timeday.GetComponent<Light>();

}

void Update()

{

intensity = sunlight.intensity;

Collider[] findedItems = Physics.OverlapSphere(transform.position, radius, itemsLayer);

k = findedItems.Length;

if (findedItems.Length > 0)

{

position = transform.position;

closest = findedItems[0];

if (k > 1)

{

cl1 = findedItems[1];

if (k > 2)

{ cl2 = findedItems[2]; }

else { cl2 = null; }

}

else

{

cl1 = null;

cl2 = null;

}

LightsOnTrigger = true;

}

else

{

LightsOnTrigger = false;

}

}

void Shadow(Collider cl, float f, SpriteRenderer \_sprite)

{

\_sprite.gameObject.SetActive(true);

scale = new Vector3(\_sprite.transform.localScale.x, \_sprite.transform.localScale.y, \_sprite.transform.localScale.z);

newscale = scale;

var color = \_sprite.color;

position = \_sprite.transform.position;

diff = cl.transform.position;

distance = Mathf.Sqrt(Mathf.Pow(position.x - diff.x, 2) + Mathf.Pow(position.y - diff.y, 2));

scalelight = Mathf.Abs(diff.z);

Debug.Log(scaleparent);

if (scalelight > scaleparent) // Изменение размера тени в зависимости где находится источник света

{

newscale = new Vector3(scale.x, scalelight \* distance / Mathf.Abs(scalelight - scaleparent) / 5, 0f);

}

else

{

newscale = new Vector3(scale.x, y \* h, 0f);

}

ang = (position.x - diff.x) / (position.y - diff.y);

if (position.y - diff.y > 0)

{

angle = -Mathf.Atan(ang) \* 180 / Mathf.PI;

}

else

{

angle = 180 - Mathf.Atan(ang) \* 180 / Mathf.PI;

}

\_sprite.transform.localScale = newscale;

\_sprite.transform.localEulerAngles = new Vector3(20, 0, angle);

color.a = (radius - distance) \* 0.4f \* f\*(1.2f-intensity);

color.a = Mathf.Clamp(color.a, 0, 0.4f);

\_sprite.color = color;

}

// Update is called once per frame

void FixedUpdate()

{

if (LightsOnTrigger)

{

if (k == 1)

{

Shadow(closest, 1, \_sprite);

\_sprite2.gameObject.SetActive(false);

\_sprite3.gameObject.SetActive(false);

}

else if (k == 2)

{

Shadow(closest, 0.9f, \_sprite);

Shadow(cl1, 0.9f, \_sprite2);

\_sprite3.gameObject.SetActive(false);

}

else if (k >= 3)

{

Shadow(closest, 0.8f, \_sprite);

Shadow(cl1, 0.8f, \_sprite2);

Shadow(cl2, 0.8f, \_sprite3);

}

}

else

{

\_sprite.gameObject.SetActive(false);

\_sprite2.gameObject.SetActive(false);

\_sprite3.gameObject.SetActive(false);

}

}

}