МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Российский государственный университет им. А. Н. Косыгина   
(Технологии. Дизайн. Искусство)»

Кафедра информационных технологий

**Курсовая работа**

по дисциплине «Объектно-ориентированное программирование»

по теме: «Создание виртуального химического движка для будущего использования при создании игр»

Выполнил:

студент 2 курса Кузнецов А.А.   
группа МВС–122 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*подпись*

Проверил:   
к.т.н., доцент Семенов А.А.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*подпись*

Дата «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ г.

РГУ им. А.Н. Косыгина, 2024 г.

**Содержание**

[Введение 3](#_gjdgxs)

**Постановка задачи 3**

[Обоснование выбора программных средств для решения задачи 3](#_1fob9te)

[Описание использованных средств языка программирования 3](#_3znysh7)

**Руководство пользователя 4**

**Листинг кода 7**

[Chemical\_Material 7](#_2et92p0)

[ParentOfMaterials 8](#_tyjcwt)

[Выводы 10](#_3dy6vkm)

[Список литературы 11](#_1t3h5sf)

# Введение

В современной разработке игр есть множество готовых движков и дополнений к движку, которые позволяют сократить время разработки, а соответственно и деньги. Химический движок одно из немногих вещей в разработке игр, которых нет в общем доступе, несмотря на то что проекты где они используются сильно выделяются на фоне остальных игр. “The legend of Zelda”, “Noita” это два очень известных и успешных проекта в которых использовались химические движки. Это показывает актуальность для пользователя химических движков в игровых проектах.

# Постановка задачи

Целью данной работы сделать универсальную и гибкую основу для химического движка которая позволит делать его в 2D и 3D проектов, с любой оптимизацией, любыми способами взаимодействия элементов игры, которые отличаются от игры к игре.

# Обоснование выбора программных средств для решения задачи

Выбор игрового движка Unity для разработки данного движка обусловлен его ориентированностью на компонентном программировании, а также относительно простым переходом между 3D и 2D.C# был выбран в качестве языка программирования по причине того, что он является основным языком в Unity.

# Описание использованных средств языка программирования

1. Классы и объекты – это, основные конструкции объектно-ориентированного подхода в С#. Классы представляют собой шаблоны, по которым создаются объекты. Они содержат поля (данные) и методы (функции), которые определяют поведение объектов.

2. Компоненты - В Unity, компоненты – это модульные блоки, которые добавляются к игровым объектам для придания им функциональности. Компоненты взаимодействуют друг с другом и с игровым движком, определяя поведение игрового объекта.

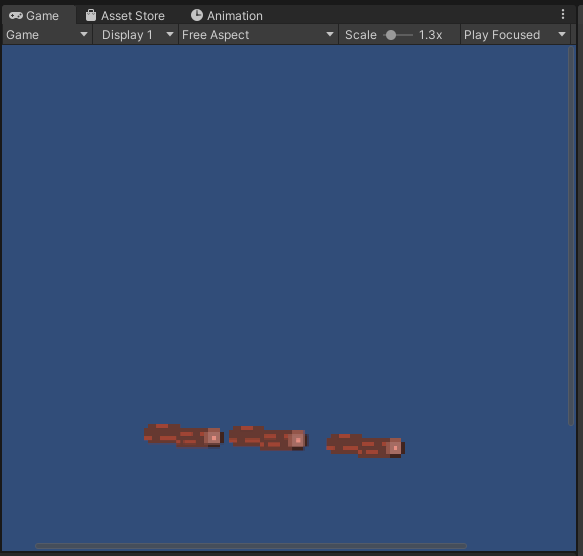
3. Методы - Функции, определенные внутри класса, которые выполняют действия с данными объекта или взаимодействуют с другими объектами. Могут быть как публичными, так и приватными

4. Стандартные методы – это, функции, встроенные в движок Unity, они позволяют использовать уже готовую логику или активировать логику от игровых событий.

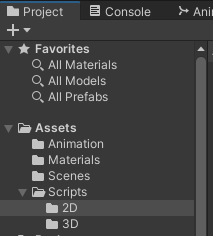
5. Наследование - Механизм, позволяющий создавать новые классы на основе существующих классов. Производные классы наследуют поля и методы базового класса и могут добавлять свои собственные.

# Руководство пользователя

При открытии проекта пользователя встречает сцена, где представлен стартовый функционал движка.



А также папка ассетов для создания собственного движка и этой сцены.

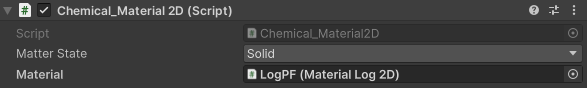


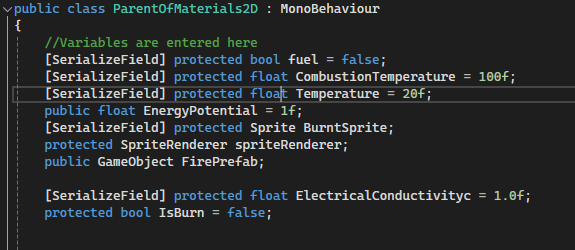
В папке 2D и 3D находятся скрипты для реализации движка:

1)Chemical\_Material(Chemical\_Material2D) - это скрипт для оптимизации химических элементов под конкретный проект он выполняет функцию активатора логики в следующем скрипте.

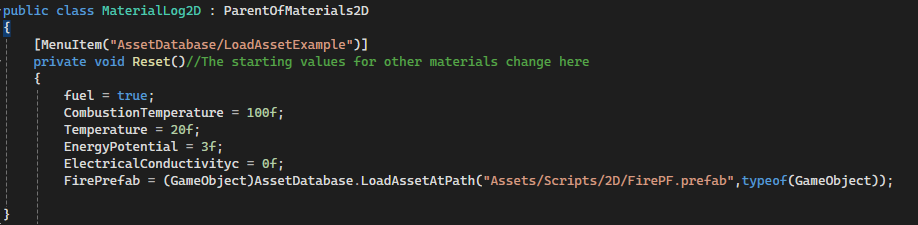
2)ParentOfMaterials(ParentOfMaterials2D) - это скрипт в котором находится вся логика взаимодействия объекта а также его характеристики.

Чтобы добавить логику материала необходимо добавить через inspector объекту Chemical\_Material(Chemical\_Material2D) и ParentOfMaterials(ParentOfMaterials2D)/класс наследник. А также перетащить материал в Chemical\_Material(Chemical\_Material2D).Material.

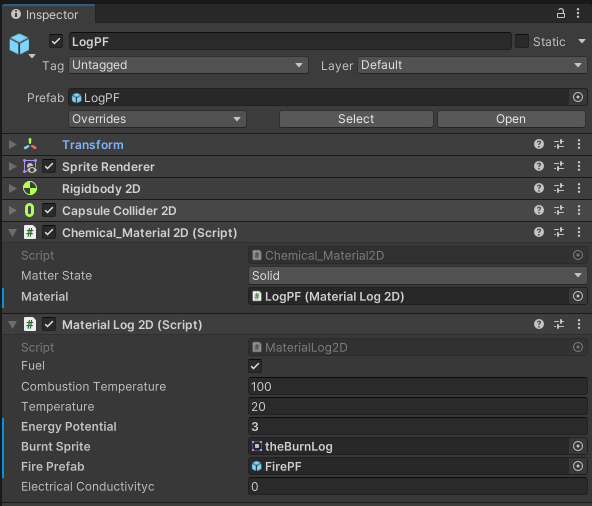


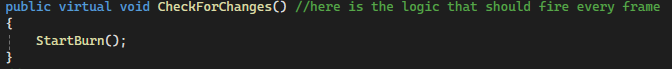
Чтобы добавить новую характеристику всем материалам, необходимо открыть ParentOfMaterials(ParentOfMaterials2D) и добавить ввод переменной до всей логики 

Чтобы добавить уникальную для материала характеристику то необходимо делать ввод уже в самом классе наследнике.

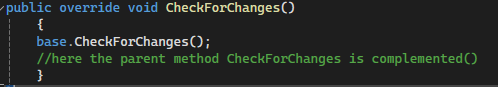
Чтобы изменить стартовые характеристики класса наследника, нужно задать значение унаследованной переменной в методе Reset:

Корректировку значений можно осуществить напрямую через сцену в inspector



Дополнение логики родительского материала в классах наследниках осуществляется через виртуальный метод в родительском классе:

И override и base в классе наследнике:



# Листинг кода

В ходе работы я использовал 2 формы. Первая форма используется для управления. Следующий код относится к ней.

## Chemical\_Material

using UnityEngine;

public class Chemical\_Material : MonoBehaviour

{

//here the choice of the form of matter is made

public enum StateOfMatter

{

Liquid,

Gas,

Solid

}

//Variables are entered here

[SerializeField] StateOfMatter MatterState;

[SerializeField] ParentOfMaterials Material;

Component \_material;

void Awake()

{

}

void FixedUpdate()//Here logic is implemented for each form of matter

{

switch (MatterState)

{

case StateOfMatter.Liquid:

break;

case StateOfMatter.Gas:

break;

case StateOfMatter.Solid:

if (Material != null)

{

Material.CheckForChanges();

}

break;

}

}

}

## ParentOfMaterials

using UnityEngine;

public class ParentOfMaterials: MonoBehaviour

{

//Variables are entered here

[SerializeField] bool fuel = false;

[SerializeField] float CombustionTemperature = 100f;

[SerializeField] float Temperature = 20f;

[SerializeField] float EnergyPotential = 0f;

[SerializeField] float BurningRate = 1f;

[SerializeField] float ElectricalConductivityc = 1.0f;

protected bool IsBurn = false;

public void CheckForChanges() //here is the logic that should fire every frame

{

}

//Next we write your methods for simulating your chemical processes.

}

# Выводы

Unity- гибкий инструмент разработки игр с огромным количеством инструментов и методов, что позволяет реализовывать любой игровой проект, однако при работе с Unity, есть свои правила и стандарты которые нужно учитывать в работе над проектом.

# Список литературы

1. “C# 2024 С НУЛЯ ДО ПРОФИ | СЛИВ ЛУЧШЕГО КУРСА“ // YouTube URL: https://www.youtube.com/watch?v=w8rRhAup4kg (дата обращения: 07.02.2024).
2. “Unity Documentation https://docs.unity3d.com/ScriptReference/MonoBehaviour.Reset.html