# **Документация по игровому движку PEngenAPI**

Данная документация описывает использование компонентов игрового движка, реализованных в файлах GnuEngenSFML.cs, Class1.cs (GameEngine) и SuperUpdate.cs.

## **1. Обзор архитектуры**

Движок состоит из трех основных частей:

1. **GnuEngenSFML**: Графический движок, основанный на SFML, отвечающий за отрисовку 2D-объектов, обработку ввода (мышь, текст) и анимацию.
2. **GameEngine**: Физический и логический движок, отвечающий за управление объектами (GameObject), их физическое взаимодействие (столкновения, перемещение), и выполнение логики обновления.
3. **SuperUpdate**: Компонент-обертка, предназначенный для унифицированного управления объектами из обоих движков (GnuEngenSFML и GameEngine) как единым целым, используя структуру GameOBJN.

## **2. Использование GameEngine (Физика и Логика)**

Класс PEngenAPI.GameEngine является центральным для управления игровой логикой и физикой.

### **2.1. Класс GameEngine.GameObject**

Это основной объект, с которым работает физический движок.

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| x, y | float | Координаты объекта. Обновляются физическим движком. |
| vx, vy | float | Скорость объекта по осям X и Y. |
| width, height | float | Размеры объекта для расчета коллизий (AABB). |
| isStatic | bool | Если true, объект не перемещается физическим движком и используется как статическая преграда. |
| mass | float | Масса объекта (по умолчанию 1.0f). Используется при расчете столкновений. |
| OnUpdate | Action<GameObject> | Делегат, вызываемый каждый кадр для выполнения игровой логики. |
| visual | GnuEngenSFML.D2Gun | Визуальное представление объекта, которое будет отображено графическим движком. |

### **2.2. Основные методы GameEngine**

|  |  |
| --- | --- |
| Метод | Описание |
| GameEngine() | **Конструктор.** Инициализирует графический движок (GnuEngenSFML) и создает окно. |
| AddObject(GameObject obj) | Добавляет объект в физический движок и, если есть obj.visual, в графический движок. |
| DeleteObject(GameObject obj) | Удаляет объект из движков. |
| UpdateObject(GameObject oldObj, GameObject newObj) | Обновляет существующий объект oldObj на newObj в списке объектов движка. **Внимание:** ищет по ссылке oldObj. |
| Run() | **Основной цикл игры.** Запускает постоянный цикл: UpdatePhysics, HandleCollisions, UpdateLogic, graphicsEngine.Draw. |

### **2.3. Пример использования GameEngine**

C#

using PEngenAPI; using static PEngenAPI.GnuEngen.GnuEngenSFML; // Для доступа к D2Gun// 1. Создание экземпляра движка GameEngine engine = new GameEngine(); // 2. Создание визуального представления (D2Gun) D2Gun visualBall = new D2Gun { x = 100, y = 100, sx = 50, sy = 50, type = ObjectType.Cube, // Куб для простоты base\_color = 0x00FF00 // Зеленый }; // 3. Создание игрового объекта (GameObject) GameEngine.GameObject ball = new GameEngine.GameObject { x = 100, y = 100, width = 50, height = 50, vx = 50f, vy = 100f, // Начальная скорость mass = 5.0f, visual = visualBall, // Добавление логики: падение (простая гравитация) OnUpdate = (gameObject) => { // Увеличение вертикальной скорости с течением времени (гравитация) gameObject.vy += 9.8f \* 0.1f; // (9.8 м/с^2 \* dt) } }; // 4. Создание статического объекта (Стена) D2Gun visualWall = new D2Gun { x = 50, y = 500, sx = 700, sy = 50, type = ObjectType.Cube, base\_color = 0xFF0000 // Красный }; GameEngine.GameObject wall = new GameEngine.GameObject { x = 50, y = 500, width = 700, height = 50, isStatic = true, // Статичный объект, не будет двигаться visual = visualWall }; // 5. Добавление объектов в движок engine.AddObject(ball); engine.AddObject(wall); // 6. Запуск игрового цикла engine.Run();

## **3. Использование GnuEngenSFML (Графика и UI)**

Класс PEngenAPI.GnuEngen.GnuEngenSFML управляет графикой и элементами пользовательского интерфейса (UI). **Обычно вы взаимодействуете с ним через GameEngine или SuperUpdate**, но можете использовать его напрямую для UI-элементов.

### **3.1. Класс GnuEngenSFML.D2Gun**

Это универсальный класс для всех 2D-объектов, которые может отрисовать графический движок.

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| x, y, sx, sy | uint | Координаты (x, y) и размеры (sx - ширина, sy - высота). |
| type | ObjectType | Тип объекта (Cube, Line, Sprite, Text, Button, Input и т.д.). |
| base\_color | uint | Цвет в формате 0xRRGGBB. |
| texture | Texture | Текстура для Sprite или Button. |
| animation | Animation | Объект анимации для циклических кадров. |
| text | string | Текст для Text, TextButton. |
| font | Font | Шрифт для текста. |
| onClick | Action | Действие, выполняемое при нажатии на Button или TextButton. |
| inputString | string | Содержимое для Input поля. |
| isActive | bool | Указывает, активно ли поле Input для ввода текста. |

### **3.2. Типы объектов (ObjectType)**

Перечисление GnuEngenSFML.ObjectType определяет, как будет отрисовываться D2Gun:

* Cube: Простой прямоугольник с цветом.
* Line: Отрезок.
* **Sprite**: Изображение с текстурой/анимацией.
* Text: Вывод текста.
* Mesh: Четырехугольник (квадрат/прямоугольник), отрисованный как примитив.
* **Button**: Кнопка со спрайтом и обработчиком onClick.
* **TextButton**: Кнопка с прямоугольником, текстом и обработчиком onClick.
* **Input**: Поле ввода текста с поддержкой обратного вызова Window\_TextEntered.

### **3.3. Пример создания UI-кнопки**

C#

using PEngenAPI.GnuEngen; using SFML.Graphics; using System; using static PEngenAPI.GnuEngen.GnuEngenSFML; // Инициализация (обычно происходит внутри GameEngine) GnuEngenSFML gEngine = new GnuEngenSFML(); gEngine.InitWindow(800, 600, "UI Test"); // Создание функции, которая будет выполняться при нажатии Action onButtonClick = () => { Console.WriteLine("Кнопка была нажата!"); }; // Создание объекта D2Gun типа TextButton D2Gun myButton = new D2Gun { x = 300, y = 200, sx = 200, sy = 50, type = ObjectType.TextButton, text = "Нажми меня!", font = new Font("arial.ttf"), // Требуется файл шрифта onClick = onButtonClick }; // Добавление объекта в графический движок gEngine.AddObject(myButton); // Для отображения нужно вызвать Draw в цикле (GameEngine.Run() делает это автоматически)// В данном примере, если нет GameEngine.Run(), нужно реализовать цикл вручную:// while (gEngine.window.IsOpen) gEngine.Draw();

## **4. Использование SuperUpdate (Унифицированное управление)**

Класс PEngenAPI.SuperUpdate.SuperUpdate объединяет управление объектами из GnuEngenSFML и GameEngine. Это полезно для создания единого менеджера объектов сцены.

### **4.1. Структура SuperUpdate.GameOBJN**

Хранит информацию об объекте, включая его тип (Gnu для графического/UI или Engen для физического/игрового).

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| name | string | Уникальное имя для поиска и управления. |
| d2Gun | GnuEngenSFML.D2Gun | Объект графического движка (если gnuType = Gnu). |
| gameObject | GameEngine.GameObject | Объект физического движка (если gnuType = Engen). |
| gnuType | GnuType | Тип: Gnu (только графика) или Engen (физика/игра). |

### **4.2. Основные методы SuperUpdate**

|  |  |
| --- | --- |
| Метод | Описание |
| SuperUpdate(GnuEngenSFML gEngine, GameEngine engine) | **Конструктор.** Принимает ссылки на оба движка. |
| AddObject(GameOBJN obj) | Добавляет объект в список SuperUpdate и в соответствующий движок (gEngine или engine). |
| DeleteObject(string name) | Удаляет объект по имени из SuperUpdate и из соответствующего движка. |
| FindObject(string name) | Находит объект по имени. Возвращает GameOBJN? (nullable). |
| Exists(string name) | Проверяет, существует ли объект с данным именем. |
| Clear() | Удаляет все управляемые объекты из SuperUpdate и обоих движков. |
| UpdateObject(GameOBJN obj) | Обновляет объект в списке SuperUpdate по его имени. |
| Update() | Вызывает методы UpdateObject для всех объектов в обоих движках. **Внимание:** В GameEngine.Run() эта функция не вызывается автоматически, ее нужно вызывать вручную, если вы используете SuperUpdate.Update(). |

### **4.3. Пример использования SuperUpdate**

C#

using PEngenAPI; using PEngenAPI.SuperUpdate; using static PEngenAPI.GnuEngen.GnuEngenSFML; // 1. Инициализация движков GameEngine engine = new GameEngine(); GnuEngenSFML gEngine = engine.graphicsEngine; // Получаем ссылку на графический движок из GameEngine// 2. Инициализация SuperUpdate SuperUpdate superUpdater = new SuperUpdate(gEngine, engine); // 3. Создание игрового объекта (Engen Type) GameEngine.GameObject playerObj = new GameEngine.GameObject { /\* ... параметры ... \*/ }; GameOBJN player = new GameOBJN { name = "Player", gameObject = playerObj, gnuType = SuperUpdate.GnuType.Engen }; superUpdater.AddObject(player); // 4. Создание объекта UI (Gnu Type) D2Gun uiText = new D2Gun { /\* ... параметры ... \*/ }; GameOBJN scoreText = new GameOBJN { name = "ScoreDisplay", d2Gun = uiText, gnuType = SuperUpdate.GnuType.Gnu }; superUpdater.AddObject(scoreText); // 5. Запуск игры engine.Run(); // Внутри игровой логики, например, в OnUpdate другого объекта:// 6. Обновление объекта UI (например, счетчика) GameOBJN? currentScoreObj = superUpdater.FindObject("ScoreDisplay"); if (currentScoreObj.HasValue) { D2Gun updatedText = currentScoreObj.Value.d2Gun; updatedText.text = "Score: 100"; // Создаем новый объект GameOBJN для обновления GameOBJN updatedScoreObj = new GameOBJN { name = "ScoreDisplay", d2Gun = updatedText, gnuType = SuperUpdate.GnuType.Gnu // Важно сохранить тип }; superUpdater.UpdateObject(updatedScoreObj); }