# **Документация игрового движка PEngen Web**

PEngen Web — это JavaScript-движок, имитирующий структуры, похожие на SFML, с отдельными компонентами для графики (**GnuEngenSFML**) и игровой логики/физики (**GameEngine**).

## **1. Компонент GnuEngenSFML (Графика и Ввод)**

Класс GnuEngenSFML управляет окном, отрисовкой и интерактивными элементами (Спрайты, Кнопки, Текст).

### **1.1. Класс GnuEngenSFML (Отрисовка)**

|  |  |  |
| --- | --- | --- |
| Метод/Свойство | Описание | Пример использования |
| new GnuEngenSFML() | Создает экземпляр графического движка. | Program.gEngine = new GnuEngenSFML(); |
| InitWindow(w, h, title) | Инициализирует окно (Canvas). Показывает интро-экран. | gEngine.InitWindow(800, 600, "My Game"); |
| AddObject(obj) | Добавляет объект **D2Gun** в список отрисовки. | gEngine.AddObject(mySprite); |
| DeleteObject(obj) | Удаляет объект **D2Gun** из отрисовки. | gEngine.DeleteObject(intro); |
| UpdateObject(old, new) | Заменяет старый объект новым в списке отрисовки. | gEngine.UpdateObject(oldText, newText); |
| Draw() | Основная функция отрисовки. Вызывает отрисовку всех объектов, обновляет анимации, обрабатывает hover-эффекты кнопок. **Должна вызываться в цикле игры.** | requestAnimationFrame(gameLoop); // внутри gameLoop вызывается gEngine.Draw(); |
| objects | Массив (D2Gun[]) всех объектов для отрисовки. |  |

### **1.2. Класс GnuEngenSFML.D2Gun (Графический объект)**

Базовый класс для всех 2D элементов, отображаемых на экране.

|  |  |  |
| --- | --- | --- |
| Свойство | Тип | Описание |
| x, y | Число | Позиция объекта. |
| sx, sy | Число | Размеры объекта (ширина/высота). |
| type | ObjectType | **Обязательно** устанавливается тип объекта (см. ниже). |
| texture | Image | HTML-объект Image для спрайтов/кнопок без анимации. |
| animation | Animation | Объект Animation для анимированных спрайтов/кнопок. |
| base\_color | Число (Hex) | Цвет для объектов типа Cube, Line, Mesh или цвет текста (например, 0xFFFFFF). |
| text | Строка | Содержимое текста для объектов Text и TextButton. |
| font | Font | Объект Font для текста. |
| onClick | Функция | Обработчик, вызывается при клике (для Button, TextButton). |
| inputString | Строка | Текущая введенная строка (для Input). |
| isActive | Булево | Флаг активности поля ввода (для Input). |

### **1.3. Перечисление GnuEngenSFML.ObjectType (Типы объектов)**

Обязательно используйте этот статический список для указания типа вашего объекта D2Gun.

|  |  |  |
| --- | --- | --- |
| Свойство | Значение | Описание |
| **ObjectType.Cube** | 0 | Залитый цветом прямоугольник. |
| **ObjectType.Line** | 1 | Линия между (x, y) и (x+sx, y+sy). |
| **ObjectType.Sprite** | 2 | Отрисовка текстуры (texture или animation). |
| **ObjectType.Text** | 3 | Отрисовка текста (text). |
| **ObjectType.Mesh** | 4 | Залитый цветом прямоугольник (похоже на Cube). |
| **ObjectType.Button** | 5 | Кнопка-спрайт с hover-эффектом и onClick. |
| **ObjectType.TextButton** | 6 | Кнопка-прямоугольник с текстом, hover-эффектом и onClick. |
| **ObjectType.Input** | 7 | Поле ввода для текста. |

**Пример создания Cube:**

JavaScript

const redBlock = new GnuEngenSFML.D2Gun(); redBlock.x = 10; redBlock.y = 10; redBlock.sx = 50; redBlock.sy = 50; redBlock.base\_color = 0xFF0000; // Красный redBlock.type = GnuEngenSFML.ObjectType.Cube; Program.gEngine.AddObject(redBlock);

**Пример создания TextButton:**

JavaScript

const startBtn = new GnuEngenSFML.D2Gun(); startBtn.x = 100; startBtn.y = 200; startBtn.sx = 200; startBtn.sy = 50; startBtn.type = GnuEngenSFML.ObjectType.TextButton; startBtn.text = "Начать игру"; startBtn.font = Program.font; startBtn.onClick = () => Program.StartGame(); Program.gEngine.AddObject(startBtn);

### **1.4. Класс GnuEngenSFML.Animation**

Используется для анимирования объектов D2Gun.

|  |  |  |
| --- | --- | --- |
| Свойство/Метод | Тип | Описание |
| new GnuEngenSFML.Animation() | Конструктор. |  |
| Frames | Массив (Image[]) | Список объектов Image, составляющих кадры анимации. |
| FrameSpeed | Число | Время (в секундах) между кадрами (по умолчанию 0.1). |
| Update(deltaTime) | **Автоматически вызывается в gEngine.Draw().** Обновляет текущий кадр. |  |
| GetCurrentFrame() | Возвращает текущий объект Image кадра. |  |

**Пример использования анимации:**

JavaScript

const playerAnim = new GnuEngenSFML.Animation(); playerAnim.Frames.push(new Image().src = "frame1.png"); playerAnim.Frames.push(new Image().src = "frame2.png"); playerAnim.FrameSpeed = 0.2; // 5 кадров в секундуconst player = new GnuEngenSFML.D2Gun(); player.type = GnuEngenSFML.ObjectType.Sprite; player.animation = playerAnim; // Привязка анимации Program.gEngine.AddObject(player);

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

Класс GameEngine управляет игровыми объектами, их движением, физикой и столкновениями.

### **2.1. Класс GameEngine**

|  |  |  |
| --- | --- | --- |
| Метод/Свойство | Описание | Пример использования |
| new GameEngine() | Создает экземпляр игрового движка. | Program.engine = new GameEngine(); |
| AddObject(obj) | Добавляет объект **GameObject** в симуляцию. Если у него есть visual, он также добавляется в gEngine. | engine.AddObject(myPhysicsObject); |
| DeleteObject(obj) | Удаляет объект из симуляции и его visual из gEngine. | engine.DeleteObject(enemy); |
| UpdateObject(old, new) | Заменяет старый игровой объект новым в симуляции и обновляет его visual. |  |
| Run() | Запускает **основной игровой цикл**, который обрабатывает физику, коллизии, логику и вызывает gEngine.Draw(). | Program.engine.Run(); |

### **2.2. Класс GameEngine.GameObject (Физический объект)**

Класс для объектов, участвующих в физической симуляции.

|  |  |  |
| --- | --- | --- |
| Свойство | Тип | Описание |
| x, y | Число | Позиция объекта в физическом мире. |
| vx, vy | Число | Скорость по осям X и Y (velocity). |
| width, height | Число | Размеры для расчета AABB коллизий. |
| isStatic | Булево | Если true, объект не двигается физикой. **(По умолчанию false)**. |
| mass | Число | Масса объекта (влияет на отскок при столкновении). **(По умолчанию 1.0)**. |
| OnUpdate | Функция | Функция, вызываемая в цикле логики. (obj) => { ... } |
| visual | GnuEngenSFML.D2Gun | Привязанный графический объект. Его позиция обновляется движком. |

**Пример создания физического объекта:**

JavaScript

const boxVisual = new GnuEngenSFML.D2Gun(); // ... настройка boxVisual ...const boxPhysics = new GameEngine.GameObject(); boxPhysics.x = 50; boxPhysics.y = 50; boxPhysics.width = 50; boxPhysics.height = 50; boxPhysics.mass = 5.0; boxPhysics.vx = 100; // Начать движение вправо со скоростью 100 пикс/с boxPhysics.visual = boxVisual; boxPhysics.OnUpdate = (self) => { // Пользовательская логика, например, ограничение скорости }; Program.engine.AddObject(boxPhysics);

## **3. Общие Вспомогательные Классы**

### **3.1. Класс Clock (Время)**

Используется для измерения времени между кадрами (deltaTime) и создания асинхронных задержек.

|  |  |
| --- | --- |
| Метод/Свойство | Описание |
| new Clock() | Создает часы. |
| restart() | Сбрасывает часы и возвращает объект с прошедшим временем. |
| asSeconds() | Метод, который нужно вызывать на результате restart(), чтобы получить прошедшее время в секундах. |
| Clock.WaitSeconds(seconds) | **Статический асинхронный метод.** Ждет указанное количество секунд, продолжая отрисовку. Используется с await. |

**Пример таймера:**

JavaScript

const clock = new Clock(); // ... в циклеconst deltaTime = clock.restart().asSeconds(); // ...

**Пример задержки:**

JavaScript

// ... в async функцииawait Clock.WaitSeconds(3); // Ждать 3 секунды// ...

### **3.2. Классы Ввода (Keyboard и Mouse)**

|  |  |  |
| --- | --- | --- |
| Команда | Класс | Описание |
| Keyboard.isKeyPressed(key) | Keyboard | Проверяет, нажата ли клавиша (например, 'W', 'E', 'Enter'). **Важно:** используйте .toUpperCase(). |
| Mouse.GetPosition(window) | Mouse | Возвращает объект { x: num, y: num } с координатами курсора внутри окна. |
| Mouse.IsButtonPressed(button) | Mouse | Проверяет, нажата ли кнопка мыши (имитирует левую кнопку). |

### **3.3. Класс GnuEngenSFML.FloatRect (Коллизии)**

Используется для проверки прямоугольных коллизий, не зависящих от физического движка.

|  |  |
| --- | --- |
| Метод | Описание |
| new FloatRect(x, y, width, height) | Создает прямоугольник. |
| intersects(otherRect) | Возвращает true, если прямоугольник пересекается с otherRect. |
| Contains(x, y) | Возвращает true, если точка (x, y) находится внутри прямоугольника. |

## **4. Компонент SuperUpdate (Сложное Управление Объектами)**

Класс SuperUpdate предназначен для удобного управления объектами, которые могут быть либо графическими (GnuType.Gnu), либо физическими (GnuType.Engen), или и теми, и другими, объединяя их под одним именем.

### **4.1. Класс SuperUpdate**

|  |  |  |
| --- | --- | --- |
| Метод | Описание | Пример использования |
| new SuperUpdate(gEngine, engine) | Конструктор. Принимает экземпляры обоих движков. | Program.su = new SuperUpdate(gEngine, engine); |
| AddObject(obj) | Добавляет объект **GameOBJN** и его компоненты в соответствующий движок. | su.AddObject(playerObjN); |
| DeleteObject(name) | Удаляет объект с указанным именем из списка и из соответствующих движков. | su.DeleteObject("player"); |
| FindObject(name) | Находит объект **GameOBJN** по имени. | const p = su.FindObject("player"); |
| Clear() | Удаляет все объекты из SuperUpdate и обоих движков. | su.Clear(); |
| Exists(name) | Проверяет, существует ли объект с таким именем. |  |
| Update() | Обновляет все объекты в списках движков. |  |

### **4.2. Класс SuperUpdate.GameOBJN (Единый объект)**

Единая сущность, которая может содержать как графический, так и физический компонент.

|  |  |  |
| --- | --- | --- |
| Свойство | Тип | Описание |
| name | Строка | Уникальное имя объекта для поиска. |
| d2Gun | GnuEngenSFML.D2Gun | Графический компонент. |
| gameObject | GameEngine.GameObject | Физический компонент. |
| gnuType | SuperUpdate.GnuType | Тип компонента, который будет добавлен/удален: GnuType.Gnu или GnuType.Engen. |

**Пример создания и добавления объекта через SuperUpdate:**

JavaScript

// 1. Создаем графический и физический компонентыconst playerVisual = new GnuEngenSFML.D2Gun(); // ... настройка графики ...const playerPhysics = new GameEngine.GameObject(); // ... настройка физики ... playerPhysics.visual = playerVisual; // Связываем// 2. Создаем объединенный объектconst playerObjN = new SuperUpdate.GameOBJN(); playerObjN.name = "Player"; playerObjN.d2Gun = playerVisual; playerObjN.gameObject = playerPhysics; playerObjN.gnuType = SuperUpdate.GnuType.Engen; // Добавляем как физический объект, который сам добавит графику через `visual`// 3. Добавляем в SuperUpdate Program.su.AddObject(playerObjN);

Кажется, что это все основные команды и классы вашего движка! Есть ли какие-то конкретные аспекты (например, работа с полем ввода Input или физикой), которые вы хотели бы рассмотреть подробнее?