Учебный движок для создания 2D-игр "GameStudy"

Учебный движок предназначен для быстрой и легкой разработки простейших 2D-игр с использованием спрайтов, анимаций, надписей, графических примитивов и звуковых эффектов. Сама игра реализуется как программа на диалекте JavaScript путем наполнения кодом трех функций - инициализации игры, рендера игры и обработки состоянии игры.

Содержание

Структура движка

В корневом каталоге движка размещается файл GameStudy.exe, запуск которого загружает созданную игру.
Все файлы .dll и подкаталог platforms - это библиотеки, необходимые для работы движка.
Подкаталог scripts - содержит программу main.js на диалекте JavaScript (см. Приложение 1) с кодом игры.
Остальные подкаталоги содержат ресурсы игры, которые загружаются в коде.

- fonts - шрифты TTF
- sounds - звуки
- sprites - спрайты

Создание игры

Чтобы создать игру, нужно скопировать ресурсы разрабатываемой игры в соответствующие каталоги движка и наполнить функции в шаблонном файле main.js загрузкой ресурсов, рендером (выводом графики) и обработкой состояния игры.
Предопределенные функции main.js:

```
function Init() {
}
```

Функция вызывается один раз в момент старта игры, в ней нужно разместить все загрузки ресурсов и инициализацию переменных. Функция должна возвращать значение true.

```
function Render() {
}
```

Функция вызывается каждый раз, когда движок перерисовывает игровое окно, в ней нужно выводить все спрайты и надписи. Функция должна возвращать значение true.

```
function Frame(dt) {
}
```

Функция вызывается каждый раз, когда движок пересчитывает состояние игры, в ней нужно обновить состояние игровых объектов (позиции, счетчики и прочее). Параметр dt - время в секундах с момента последнего вызова функции Frame, число двойной точности. Функция должна возвращать значение true, возврат false означает остановку игры.

Простейший пример игры

В каталоге sprites размещаем файл player.png
В файле main.js размещаем код

```
// Переменную для спрайта нужно объявить в начале, чтобы её видели все функции
var spr_player ;
// То же для переменных координат спрайта
var player_x ;
var player_y ;

// Функция инициализации
function Init() {
    // Загрузка спрайта
    spr_player = game.loadSprite('player.png') ;
    // Установка координат для вывода
    player_x=400 ;
    player_y=300 ;
    return true ;
}

// Функция рендера
function Render() {
    // Выводим спрайт в заданные координаты
    spr_player.renderTo(player_x,player_y) ;
    return true ;
}

// Функция фрейма
function Frame(dt) {
    // Обрабатываем нажатия клавиш
    if (game.isKeyDown(KEY_LEFT)) player_x-=10 ;
    if (game.isKeyDown(KEY_RIGHT)) player_x+=10 ;
    if (game.isKeyDown(KEY_UP)) player_y-=10 ;
    if (game.isKeyDown(KEY_DOWN)) player_y+=10 ;
    return true ;
}
```

Данная игра выводит спрайт player.png и двигает его под действием клавиш-стрелок на клавиатуре.

Объекты движка и их функции

Глобальные объекты

В движке определены несколько глобальных объектов, методы которых вызываются из кода игры.

Объект game

Содержит функции загрузки ресурсов, создания графических примитивов, обработки нажатия клавиш и мыши.

Функции объекта game

```
loadSprite(filename)
```

Создает и возвращает объект спрайта. Параметр - имя файла из каталога sprites.

```
loadAnimationFromFiles(filenames, fps)
```

Создает и возвращает объект анимации из набора файлов. Параметры - массив имен файлов из каталога sprites, которые образуют отдельные кадры анимации, и FPS анимации. Пример вызова - loadAnimationFromFiles(['1.png','2.png','3.png',7)

```
loadAnimation(filename, w, h, framecount, fps)
```

Создает и возвращает объект анимации из одного файла, содержащего несколько кадров. Параметры - имя файла из каталога sprites, ширина и высота отдельного фрейма, число фреймов и FPS анимации. Фреймы считываются из файла слева направо, сверху вниз.

```
loadText(fontname,text,size)
```

Создает и возвращает объект надписи. Параметры - имя шрифта из каталога fonts (вместе с расширением, например, "Arial.ttf"), текст надписи, размер шрифта в надписи.

```
loadSound(filename)
```

Создает и возвращает объект звука. Параметр - имя файла из каталога sounds.

```
createLine(r,g,b)
```

Создает и возвращает объект линии. Параметры - компоненты цвета линии в RGB.

```
isKeyDown(keycode)
```

Проверяет, была ли нажата клавиша клавиатуры. Параметр - код клавиши (именованные константы клавиш приведены в Приложении 2)

```
isLeftButtonClicked()
```

Проверяет, была ли нажата левая кнопка мыши.

```
isRightButtonClicked()
```

Проверяет, была ли нажата правая кнопка мыши.

```
getMousePos()
```

Возвращает координаты мыши в игровом окне как объект с полями x и y.

```
setGameTitle(title)
```

Устанавливает заголовок окна с игрой. Параметр - строка заголовка.

```
getTotalTime()
```

Возвращает общее время с момента начала игры в секундах.

```
resetTotalTime()
```

Сбрасывает счет общего времени с момента начала игры в ноль.

```
getFPS()
```

Получает текущий FPS, усредненный за последнюю секунду.

```
setBackgroundColor(r,g,b)
```

Устанавливает цвет фона окна. Параметры - компоненты цвета текста в RGB

Объект system

Содержит функции вывода отладочного текста и отключения встроенного курсора мыши (это нужно, если игра реализует собственный курсор).

Функции объекта system

```
writeMessage(msg)
```

Выводит в текстовое поле окна отладочное сообщение. Параметр - текст сообщения.

```
showCursor(show)
```

Позволяет включить или отключить показ системного курсора в окне игры. Параметр - значение true или false.

```
saveObject(filename, value)
```

Позволяет записать в файл JSON заданный объект или массив. Параметры - имя файла и объект. Пример - мы создаём объект var state1 = { x: 1, msg:«OK», q:true }, и записывая его в файл, получим содержимое

{
"msg": "OK",
"q": true,

"x": 1
}

это позволяет сохранять данные из игры на диск.

```
loadObject(filename)
```

Считывает из файла JSON объект или массив и возвращает его. Параметр - имя файла. Работает как обратная функция для saveObject, восстанавливая объект или массив. Это позволяет как считывать ранее сохраненные данные, так и загружать в код данные из JSON-файлов, чтобы не перегружать код игры текстовыми значениями.

Загружаемые объекты ресурсов

Функции загрузки объектов создают объекты ресурсов, которые далее можно использовать в коде игры.

Объект спрайта

Содержит графический спрайт, создается методом loadSprite, может выводиться в заданное место на экране.

Функции объекта спрайта

```
renderTo(x,y)
```

Выводит спрайт со всеми установленными параметрами в окно игры. Параметры - координаты вывода спрайта относительно его начальной точки (см. функцию setHotSpot)

```
render()
```

Выводит спрайт со всеми установленными параметрами в окно игры, используя текущее положение x и y.

```
setXY(x,y)
```

Устанавливает для спрайта положение на экране. Параметры - координаты вывода спрайта относительно его начальной точки (см. функцию setHotSpot)

```
setScale(scale)
```

Устанавливает масштаб спрайта. Параметр - масштаб объекта относительно исходного в процентах.

```
setAngle(angle)
```

Устанавливает поворот спрайта. Параметр - угол поворота спрайта вокруг его начальной точки (см. функцию setHotSpot) в радианах.

```
setAlpha(alpha)
```

Устанавливает прозрачность спрайта. Параметр - значение прозрачности от 0 до 255, где 0 - это полностью прозрачный объект, 255 - полностью непрозрачный.

```
setHotSpot(x,y)
```

Устанавливает начальную точку спрайта, относительно которой работают операции вывода спрайта на экран и поворота на угол. Параметры - координаты точки в пикселях от левого верхнего угла спрайта. По умолчанию, функция loadSprite устанавливает после загрузки спрайта его точку в геометрический центр, таким образом, поворот идет сразу вокруг центра.

```
mirrorHorz(mirror)
```

Устанавливает отражение спрайта по горизонтали. Параметр - true или false, если передано true, то объект разворачивается при выводе зеркально слева направо.

```
mirrorVert(mirror)
```

Устанавливает отражение спрайта по вертикали. Параметр - true или false, если передано true, то объект разворачивается при выводе зеркально сверху вниз.

```
setBordered(bordered)
```

Включает или отключает отображение белой рамки толщиной в один пиксель вокруг спрайта. Рамка при отображении учитывает все изменения спрайта, масштабы и повороты.

```
setContactModelBox()
```

Устанавливает прямоугольную модель проверки коллизии для спрайта.

```
setContactModelCircle()
```

Устанавливает эллиптическую модель проверки коллизии для спрайта.

```
isContactWith(sprite)
```

Возвращает наличие коллизии спрайта и спрайта, переданного как параметр. Модель проверки коллизии устанавливается процедурами setContactModelBox и setContactModelCircle, по умолчанию равна прямоугольной. При использовании прямоугольной модели - коллизия возникает, если прямоугольники спрайтов пересекаются. При использовании эллиптической модели - коллизия возникает, если расстояние между центрами спрайтов меньше, чем сумма условных радиусов эллипсов, в которые вписывается спрайт. Для спрайтов визуально круглой формы (шары) целесообразно использовать эллиптическую модель, для спрайтов квадратных (коробки) - лучше работает прямоугольная модель. Обе модели учитывают все изменения спрайтов, масштабы и повороты.

```
isPointIn(x,y)
```

Возвращает вхождение точки в спрайт, с учётом всех изменений спрайтов, масштабы и повороты. Параметры - координаты проверяемой точки.

```
setTag(tagname,tagvalue)
```

Устанавливает для спрайта тэг с именем tagname в значение tagvalue. tagname должен быть строкой, а tagvalue может быть любым, включая примитивы, массивы и объекты с полями.

```
getTag(tagname)
```

Возвращает тэг с именем tagname, заданный ранее.

```
getWidth()
```

Возвращает ширину спрайта.

```
getHeight()
```

Возвращает высоту спрайта.

Объект анимации

Содержит анимацию, создается методами loadAnimationFromFiles и loadAnimation. Он поддерживает все методы объекта-спрайта, а также несколько специфичных методов для анимации

Функции объекта анимации

```
play()
```

Запускает бесконечный цикл анимации объекта.

```
playOnce()
```

Запускает однократный цикл анимации объекта, по завершении, анимация останавливается.

```
stop()
```

Останавливает цикл анимации объекта.

Объект надписи

Содержит надпись, создается методом loadText, может выводиться в заданное место на экране.

Функции объекта надписи

```
printTo(x,y)
```

Выводит надпись со всеми установленными параметрами в окно игры. Параметры - координаты вывода текста относительно его левого верхнего угла.

```
setText(text)
```

Устанавливает текст надписи. Параметр - текст. Возможно использовать в тексте Escape-последовательность \n для установки переноса текста в надписи.

```
setSize(size)
```

Устанавливает размер текста надписи. Параметр - размер.

```
setColor(r,g,b)
```

Устанавливает цвет текста надписи. Параметры - компоненты цвета текста в RGB

Объект звука

Содержит звуковой эффект, создается методом loadSound, может воспроизводиться.

Функции объекта звука

```
play()
```

Воспроизводит звуковой эффект от начала до конца, один раз. Не приводит к остановке выполнения программы на время воспроизведения, звук запускается в отдельном потоке! Также звуки могут накладываться друг на друга, если вызывать play() чаще, чем длина звука.

```
stop()
```

Останавливает воспроизведение звука.

```
isPlayed()
```

Возвращает логическое значение, соответствующее статусу звука - воспроизводится или нет в данный момент.

```
setLoop(isloop)
```

Устанавливает тип воспроизведения - если true, то звук будет крутиться в кольце. По умолчанию, звук воспроизводится только один раз.

Объект линии

Содержит линию толщиной в один пиксель, создается методом createLine, может выводиться в заданное место на экране.

Функции объекта линии

```
drawTo(x1,y1,x2,y2)
```

Выводит линию со всеми установленными параметрами в окно игры. Параметры - координаты начала и конца линии.

```
setColor(r,g,b)
```

Устанавливает цвет линии. Параметры - компоненты цвета линии в RGB

Приложение 1. Свойства и ограничения

- Размер игрового окна: 800x600 пикселей.
- Кодировка файла main.js: UTF-8
- Кодировка сохраняемых и загружаемых файлов JSON: UTF-8
- Для использования шрифта, его обязательно нужно скопировать из %WINDIR%/Fonts в каталог fonts.
- Звуковые файлы должны иметь форматы WAV или OGG. MP3 не поддерживается, рекомендуется выполнить преобразование в несжатый WAV.
- Файлы спрайтов могут иметь форматы JPG, PNG, GIF, BMP, TGA (формат GIF поддерживается как статичный - если загружаем анимированный спрайт, то будет выводиться только первый кадр)
- Язык программирования файла main.js совместим с языком JavaScript, и выполняет стандарт ECMAScript (подробнее см. на https://ru.wikipedia.org/wiki/ECMAScript)

Приложение 2. Список кодов клавиш

Для удобства разработки игры, движок определяет набор констант кодов большинства клавиш. Их можно использовать в функции isKeyDown по имени константы, или указывая непосредственное числовое значение, если так удобнее.

```
KEY_ESCAPE = 0x01000000
KEY_TAB = 0x01000001
KEY_ENTER = 0x01000005
KEY_LEFT = 0x01000012
KEY_UP = 0x01000013
KEY_RIGHT = 0x01000014
KEY_DOWN = 0x01000015
KEY_SHIFT = 0x01000020
KEY_CONTROL = 0x01000021
KEY_ALT = 0x01000023
KEY_F1 = 0x01000030
KEY_F2 = 0x01000031
KEY_F3 = 0x01000032
KEY_F4 = 0x01000033
KEY_F5 = 0x01000034
KEY_F6 = 0x01000035
KEY_F7 = 0x01000036
KEY_F8 = 0x01000037
KEY_F9 = 0x01000038
KEY_F10 = 0x01000039
KEY_F11 = 0x0100003a
KEY_F12 = 0x0100003b
KEY_SPACE = 0x20
KEY_PLUS = 0x2b
KEY_MINUS = 0x2d
KEY_0 = 0x30
KEY_1 = 0x31
KEY_2 = 0x32
KEY_3 = 0x33
KEY_4 = 0x34
KEY_5 = 0x35
KEY_6 = 0x36
KEY_7 = 0x37
KEY_8 = 0x38
KEY_9 = 0x39
KEY_A = 0x41
KEY_B = 0x42
KEY_C = 0x43
KEY_D = 0x44
KEY_E = 0x45
KEY_F = 0x46
KEY_G = 0x47
KEY_H = 0x48
KEY_I = 0x49
KEY_J = 0x4a
KEY_K = 0x4b
KEY_L = 0x4c
KEY_M = 0x4d
KEY_N = 0x4e
KEY_O = 0x4f
KEY_P = 0x50
KEY_Q = 0x51
KEY_R = 0x52
KEY_S = 0x53
KEY_T = 0x54
KEY_U = 0x55
KEY_V = 0x56
KEY_W = 0x57
KEY_X = 0x58
KEY_Y = 0x59
KEY_Z = 0x5a
```