

LUDUM

How To Use Game Engine

March 6, 2019

Contents

1	Overview	5
1.1	Install Ludum game engine	5
1.1.1	Requirements	5
1.1.2	Installation	5
2	Components	7
2.1	Entity	7
2.2	Body	8
2.3	CollisionDetection	8
2.4	Physics	9
2.5	AudioManager	9
2.6	Sprite	9
2.7	ResourceManager	9
2.8	Background	10
2.9	HUD	10
2.10	Menu	10
2.11	ScoreBoard	10
2.12	Logger	11

Chapter 1

Overview

1.1 Install Ludum game engine

1.1.1 Requirements

- vscode
- nodejs
- npm or yarn
- jest

1.1.2 Installation

Open terminal in vscode. Go to project folder via terminal and type:

- npm i
- npm start

Chapter 2

Components

2.1 Entity

All objects that the user wants to create get created via the Entity class.

The constructor of Entity takes these arguments:

- name
- body
- physics
- collisionDetection
- audioManager
- sprite

Example of how to create a *Entity*:

```
1 class Pipe {
2     constructor(startPos, topPos, height, width) {
3         this.len;
4         this.entity = new Entity(
5             "Bottom pipe",
6             new Body(this, 1920 + startPos, topPos, height, width),
7             new Physics(this, -8.85, 0),
8             new CollisionDetection(this),
9             null
10        );
11    }
12 }
```

2.2 Body

Body class is the body of the entity.

The constructor of Body takes these arguments:

- entity
- left
- top
- height
- width

The body class contains only setters and getters for these parameters.

```
1 class Bird {  
2   constructor() {  
3     this.entity = new Entity(  
4       "Bird",  
5       new Body(this, 300, 540, 100, 100),  
6     )  
7   }
```

Here is a small example of how to move the entity bird:

```
1  
2 if (this.getBody().getTop() > 1040) {  
3   this.getBody().setTop(400);  
4   this.getBody().setLeft(300);  
5 }
```

2.3 CollisionDetection

To check for collisionDetection use:

```
1 checkForCollision(otherEntity)
```

Example of this can be:

```
1  
2 let hasPlayerCollided = player  
3   .getCollisionDetection()  
4   .checkForCollision(this.state.playerArr[index].  
   getEntity());
```


2.4 Physics

2.5 AudioManager

example to use AudioManager:

Object:

```

1  new AudioManager ([
2      ResMan.getAudioPath("soundEffect1.mp3"),
3      ResMan.getAudioPath("soundEffect2.mp3"),
4      ResMan.getAudioPath("soundEffect3.mp3")
5  ]),
6  this.enum = {
7      BIRD_JUMPS: 0,
8      BIRD_SCORE: 1,
9      BIRD_DIES: 2
10 };

```

Then if something happens:

```

1
2 object.getAudioManager().play(2); // testing !!!

```

2.6 Sprite

2.7 ResourceManager

To use the ResourceManager simply import the class and use it like this:

```

1 ResourceManager.getImagePath("background.png")

```

```

1 ResourceManager.getAudioPath("one.mp3")

```

```

1 ResourceManager.getSpritePath("birds.png")

```

The *ResourceManager* uses a default paths:

- ../resources/image/
- ../resources/audio/
- ../resources/sprite/

You can change these paths in the *resourceManager* file

2.8 Background

To use the *Background* component, simply add it to the component.

```

1 <Background
2     height={1080}
3     width={1920}
4     speed={0.5}
5     image={ResourceManager.getImagePath("background.png
6     ")}
7     >
8     {" "}
    </Background>{" "}
```

ackground contains *defaultProps*, so it is not needed to set *height*, *width* and *speed*. To set a image you can either use the *ResourceManager* or simply import a image.

2.9 HUD

To use the HUD simply add the HUD component

```

1 <HUD score={this.state.score} position={"tc"} />{" "}
```

Where *score* is a score variable from the game.

2.10 Menu

To use the menu, simply import the component into the file you want.

```

1 <Menu showMenu={this.state.showMenu}
```

Using *this.state.showMenu* gives you the option of toggling it on and off, depending of a boolean *showMenu*.

To add more menu options, simply add more items into the *menuItems*, and use *handleClick(e)* for the event.

2.11 ScoreBoard

To use scoreboard simply add:

```

1 <ScoreBoard />
```

To get the score from the game, *context* is recommended, as it is shown in the *flappy* demo, since normally a *menuItem* shows a scoreboard.

2.12 Logger

To use the logger simply use:

```
1 Logger.setText("flappy.js", 'score: ${this.state.score}');
```

where first argument is the name of file and second argument is what you want to log.

Then you can add this to the game:

```
1 <LoggerManager />
```