Ludum

How To Use Game Engine

March 6, 2019

Contents

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*:

```
class Pipe {
  constructor(startPos, topPos, height, width) {
    this.len;
    this.entity = new Entity(
      "Bottom pipe",
      new Body(this, 1920 + startPos, topPos, height, width),
      new Physics(this, -8.85, 0),
      new CollisionDetection(this),
      null
    );
  }
}
```

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.

```
class Bird {
  constructor() {
    this.entity = new Entity(
      "Bird",
      new Body(this, 300, 540, 100, 100),
      }
}
```

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

```

if (this.getBody().getTop() > 1040) {
      this.getBody().setTop(400);
      this.getBody().setLeft(300);
    }
```

2.3 CollisionDetection

To check for collitionDectection use:

```
  checkForCollision(otherEntity)
```

Example of this can be:

```

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

2.4 Physics

2.5 AudioManager

example to use audioManager:
Object:

```
    new AudioManager([
        ResMan.getAudioPath("soundEffect1.mp3"),
        ResMan.getAudioPath("soundEffect2.mp3"),
        ResMan.getAudioPath("soundEffect3.mp3")
      ]),
    this.enum = {
      BIRD_JUMPS: 0,
      BIRD_SCORE: 1,
      BIRD_DIES: 2
};
```

Then if something happens:

```

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

2.6 Sprite

2.7 ResourceManager

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

```
ResourceManager.getImagePath("background.png")
```

```
ResourceManager.getAudioPath("one.mp3")
```

```
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.

```
<Background
                height={1080}
                width={1920}
                speed={0.5}
                image={ResourceManager.getImagePath("background.png
    ")}
            >
                {" "}
            </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

```
<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.

```
<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:

```
<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:

```
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:

```
<LoggerManager />
```