Ludum

How To Use Game Engine

March 7, 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
- imageRender

Example of how to create a *Entity*:

```
class Box {
  constructor(x, y, height, width) {
    this.entity = new Entity(
      "Ludum",
      new Body(this, x, y, height, width),
      new Physics(this, 10, -6),
      new CollisionDetection(this),
      null,
      null,
      new ImageRender(this, ResourceManager.getImagePath("logo.
    png"))
    );
```

```
12   }
13 }
```

Entity has these methods. These methods return either the component or null.

- getEntity() returns entity
- getName() returns name of entity as a string
- getBody() returns body
- getPhysics() return physics
- getCollisionDetection() return CollisionDetection
- getAudioManager() returns audioManager
- getSprite() returns sprite
- getImgae() returns image
- getUpdate() updates body of entity through physics
- getEntityProps() returns the newest value in body and physics

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 Object {
  constructor() {
    this.entity = new Entity(
      "Object",
      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

Physics takes 3 arguments:

- entity
- left
- top

Physics has setters for these items, while getters for left and top.

Physics also has a update arrow function. This arrow function is used to update the physics of the object from old to new position.

This method is used in Entity update.

Example:

```
 placeholder[0].getPhysics()
.setTop(this.state.playerArr[0].getPhysics().getTop() * -1);
```

This example is about changing the vertical direction of an object with 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

The Sprite component has these variables:

- entity
- spriteSheet
- rows
- columns
- spriteHeight
- spriteWidth
- speed

Example how to use a 2x4 spritesheet in a entity:

```
new Sprite(this, ResourceManager.getSpritePath("birds.png"), 2, 4, 75, 75, 12)
```

To get this to work, you need to add:

```

getSprite() {
    return this.entity.getSprite();
  }
  // rendering this class
  // rendering this class
  render() {
    return this.getSprite().render(); // rendering sprite animation
  }
```

To the object file.

2.7 ImageRender

To use ImageRender, simply add:

```
new ImageRender(this, ResourceManager.getImagePath("logo.png"))
```

To an entity.
Example:

```
class Box {
  constructor(x, y, height, width) {
    this.entity = new Entity(
      "Ludum",
      new Body(this, x, y, height, width),
      new Physics(this, 10, -6),
      new CollisionDetection(this),
      null,
      null,
      new ImageRender(this, ResourceManager.getImagePath("logo.png"))
    );
  }
```

To get this to work, you need to add:

```
1
2  getImage() {
3     return this.entity.getImage();
4   }
5   // rendering this class
6   render() {
7     return <span className="frame">{this.getImage().render()}</
     span>;
8   }
```

To the object file.

2.8 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

2.9 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.

2.10 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.

2.11 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.

2.12 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.

2.13 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 />
```

Chapter 3

More Information

To learn more about the game engine visit our github projects:
github.com/bergurijohansen/LudumGameEngine
github.com/bergurijohansen/LudumFlappy
github.com/bergurijohansen/LudumDino

3.1 Contact

Contact information:
Herborg Kristoffersen
Bergur I. Johansen
Helgi Poulsen
Lív Olsen
Fróði H. Joensen