NOTE: **TextureManager.h** in my GameEngine is the same as **Constants.h** from your description.

Game Engine Basic Instructions

- Create an Image/Sprite

Image::getInstance takes the name of the image you want to show on the screen.

```
int main(int argc, char** argv) {
    Session ses;
    Image* player = Image::getInstance("player.png");
    ses.add(player);
    ses.run();
```

- Create an Image/Sprite with BoxCollider

BoxCollider will automatically resize it self so that the element/s fits inside it

```
int main(int argc, char** argv) {
    Session ses;
    Image* player = Image::getInstance("player.png");
    BoxCollider* boxCollider = BoxCollider::getInstance();
    boxCollider->add(player);
    ses.add(player);
    ses.run();
```

- Make Image/Sprite move

Using **ses.update** to update variables values such as the players position.
NOTE: make sure you move the **BoxCollider** and not the actual Image. The **BoxCollider** will automatically move the Components/items inside it with it.
NOTE: you do not need a **BoxCollider** in order to move your **Image**.

```
int main(int argc, char** argv) {
    Session ses;
    Image* player = Image::getInstance("player.png");
    BoxCollider* boxCollider = BoxCollider::getInstance();
    boxCollider->add(player);
    ses.add(player);
    ses.update([&boxCollider/*put the address of the vairable you want to reference*/] {
        Position posTemp(boxCollider->getPosition().x + 1, boxCollider->getPosition().y);
        boxCollider->setPosition(posTemp);
        });
    ses.run();
    return 0;
}
```

- Collision

In order for collision to occur you need two objects with **BoxCollider** to collide with each other. When that happens you can simply **ignore it** or to write a **onCollisionEnter** function where you decide what to do. You get to know who did you collide with using "**other**".

```
boxCollider->onCollisionEnter([&canMove](BoxCollider* other) {
    if (other->getName() == "wall") canMove = false;
    });
```

- Input

Using **Session** you have the choice of doing something when the user enters a specific key. In the case of **KeyBoard** you need to specify which **keyboard key** you want to use. In the case of the **Mouse** you need to specify which **mouse button** you want to use + you get to know the **position** of where the click happens.

```
ses.addKeyDownCallback(SDLK_RIGHT, [&moveRight, &canMoveLeft] {
    moveRight = true;
    canMoveLeft = true;
    });
ses.addKeyUpCallback(SDLK_RIGHT, [&moveRight] {
    moveRight = false;
    });
ses.addKeyDownCallback(SDLK_LEFT, [&moveLeft, &canMoveRight] {
    moveLeft = true;
    canMoveRight = true;
    });
ses.addKeyUpCallback(SDLK_LEFT, [&moveLeft] {
    moveLeft = false;
    });
ses.addMouseDownButtonCallback(SDL_BUTTON_LEFT, [](int x, int y) {
    std::cout << "SHOOT!";
    });
```

- Change Directory for images and fonts by going to TextureManager and change them there.

```
class TextureManager {
public:
    virtual ~TextureManager();
    const SDL_Texture* getTexture();
protected:
    virtual void createTexture() = 0;
    SDL_Texture* texture = nullptr;
    std::string imagesAddress = "C:/Development/GameEngine 2/resources/images/";
    std::string fontsAddress = "C:/Development/GameEngine 2/resources/fonts/";
};
```

Game Instructions

NOTE: there is only one script for the actual game it is **MyGame.CPP**

- The player can go right or left using the **arrow keys** right and left.
- The player can shoot bullets with the **spacebar** keyboard key.
- The enemy spawns at a random position from the top of the screen. The enemy keeps falling down.
- If the enemy touches you, you lose and the application closes.