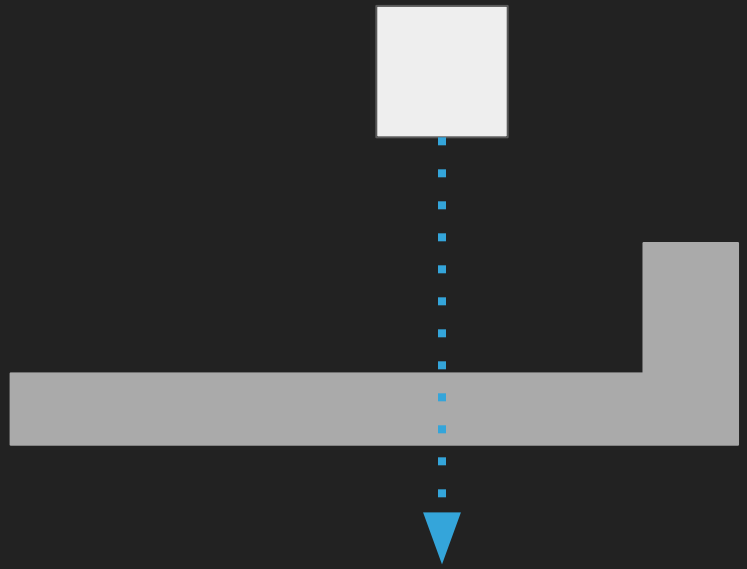
Basic Game Physics

Part 2

We need to fix some stuff!

(and add some more stuff)

Use Y velocity first...

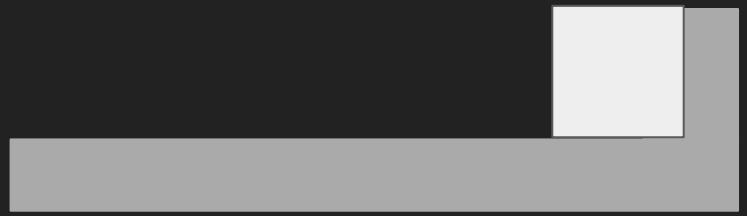


Check for collisions...

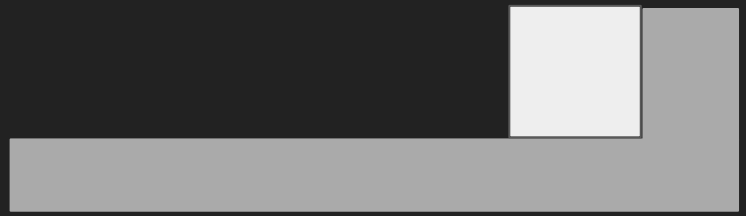
Adjust based on penetration...

Use X velocity next...

Use X velocity next...



Adjust based on penetration.



Update Code

```
void Entity::Update(float deltaTime, Entity *objects, int objectCount)
{
    velocity += acceleration * deltaTime;

    position.y += velocity.y * deltaTime;      // Move on Y
    CheckCollisionsY(objects, objectCount);    // Fix if needed

    position.x += velocity.x * deltaTime;      // Move on X
    CheckCollisionsX(objects, objectCount);    // Fix if needed
}
```

```
void Entity::CheckCollisionsY(Entity *objects, int objectCount)
{
    for (int i = 0; i < objectCount; i++)
    {
        Entity object = objects[i];

        if (CheckCollision(object))
        {
            float ydist = fabs(position.y - object.position.y);
            float penetrationY = fabs(ydist - (height / 2) - (object.height / 2));
            if (velocity.y > 0) {
                position.y -= penetrationY;
                velocity.y = 0;
            }
            else if (velocity.y < 0) {
                position.y += penetrationY;
                velocity.y = 0;
            }
        }
    }
}
```

```
void Entity::CheckCollisionsX(Entity *objects, int objectCount)
{
    for (int i = 0; i < objectCount; i++)
    {
        Entity object = objects[i];

        if (CheckCollision(object))
        {
            float xdist = fabs(position.x - object.position.x);
            float penetrationX = fabs(xdist - (width / 2) - (object.width / 2));
            if (velocity.x > 0) {
                position.x -= penetrationX;
                velocity.x = 0;
            }
            else if (velocity.x < 0) {
                position.x += penetrationX;
                velocity.x = 0;
            }
        }
    }
}
```

Let’s Code!

CheckCollisionY
CheckCollisionX
Update

Adding More to Entities

Entity Type

Entity Type

```
enum EntityType { PLAYER, PLATFORM, COIN, ENEMY };

class Entity {
public:

    EntityType entityType;

    glm::vec3 position;
    glm::vec3 velocity;
    glm::vec3 acceleration;
```

Entity Type and Update

```
void Entity::Update(float deltaTime, Entity *objects, int objectCount)
{
    if (entityType == WALL) {
        return;
    }
    else if (entityType == COIN) {
        // spin
    }
    else if (entityType == ENEMY) {
        // Move left to right
    }
    else if (entityType == PLAYER) {
        // Do all the things
    }
}
```

Entity Type and Collision

```
bool Entity::CheckCollision(Entity other)
{
    float xdist = fabs(position.x - other.position.x) - ((width + other.width) / 2.0f);
    float ydist = fabs(position.y - other.position.y) - ((height + other.height) / 2.0f);

    if (xdist < 0 && ydist < 0)
    {
        lastCollision = other.entityType;
        return true;
    }

    return false;
}
```

Entity Type and Collision

```
// Somewhere in your code

if (player.lastCollision == COIN) {
    // get points
}

else if (player.lastCollision == ENEMY) {
    // take damage
}
```

Dynamic vs. Static

isStatic

```
class Entity {
public:

    EntityType entityType;
    bool isStatic;

    glm::vec3 position;
    glm::vec3 velocity;
    glm::vec3 acceleration;
```

isStatic

(Walls, platforms, etc.)

Update

Does not move.

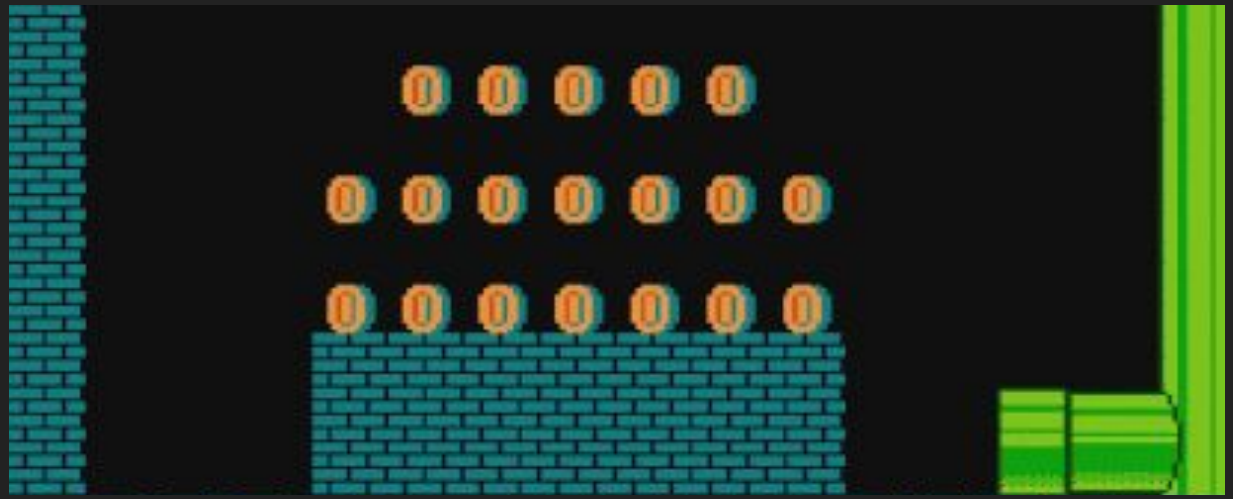
CheckCollision

Always false. Things collide with it, not it collides with things.

Render

Render as you normally would.

isActive



isActive

```
class Entity {
public:

    EntityType entityType;
    bool isStatic;
    bool isActive;

    glm::vec3 position;
    glm::vec3 velocity;
    glm::vec3 acceleration;
```

isActive

(collected coins, squashed enemies, objects in object pool)

Update

Exit right away.

CheckCollision

Always false for both objects!

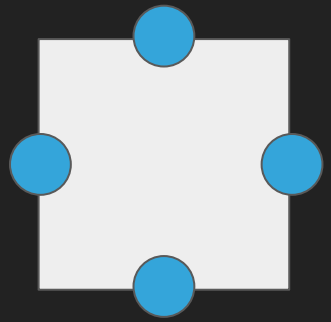
```
if (isActive == false || other.isActive == false) return false;
```

Render

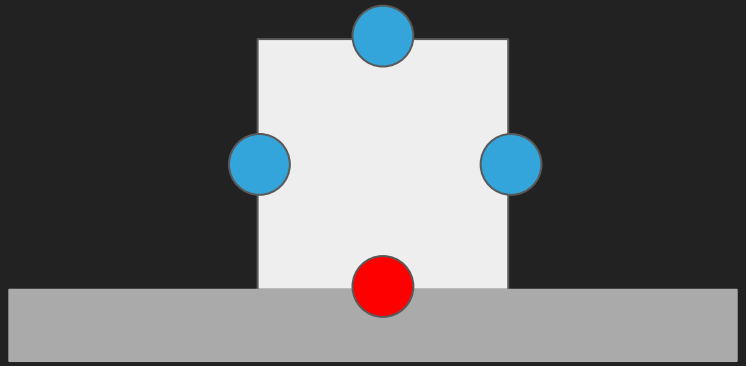
No rendering. Exit right away.

One More Thing...

Collision Flags

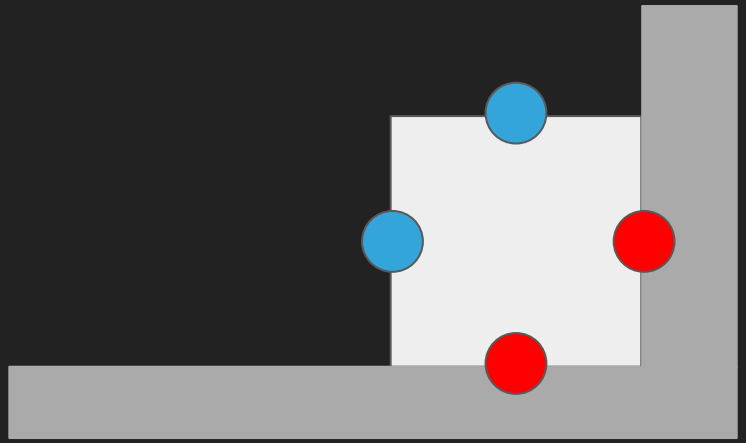


Collision Flags



Player should only be able to
jump when touching the ground.

Collision Flags



Enemies change direction after hitting a wall.

Collision Flags

```
class Entity {
public:

    bool collidedTop;
    bool collidedBottom;
    bool collidedLeft;
    bool collidedRight;
```

Project 3: Lunar Lander

Player should fall with gravity (make it very low so it moves slowly).
Moving left and right should change acceleration instead of velocity.
If the player touches a wall/rock show text “Mission Failed”
If the player touches the platform show text “Mission Successful”

You can use whatever graphics/theme as long as you meet the requirements.

Commit your code to your GitHub repository.
Post the link in the Assignments area.

For example, your link might look like:

https://github.com/tonystark/CS3113/P2/

Before we code...

It's my birthday this weekend,
project is due Tuesday Midnight!

No class Monday… was moved to Tuesday, but I'm not available Tuesday.
I will send an announcement.
(to get music and sound effects)

Let’s Code!

isStatic
isActive
Collision Flags