Basic Game Physics

Gravity

Jumping

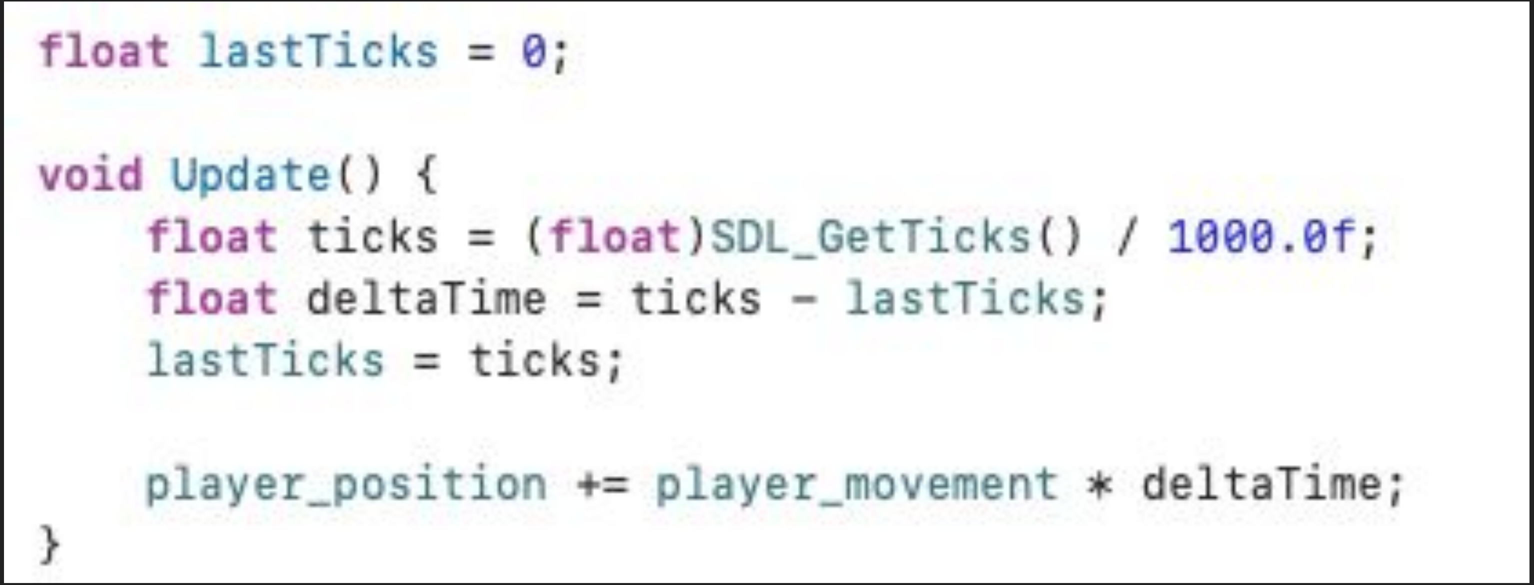
Movement

(somewhat automagically)

Topics:
Fixed Timestep
Velocity
Acceleration (Gravity)

Fixed Timestep

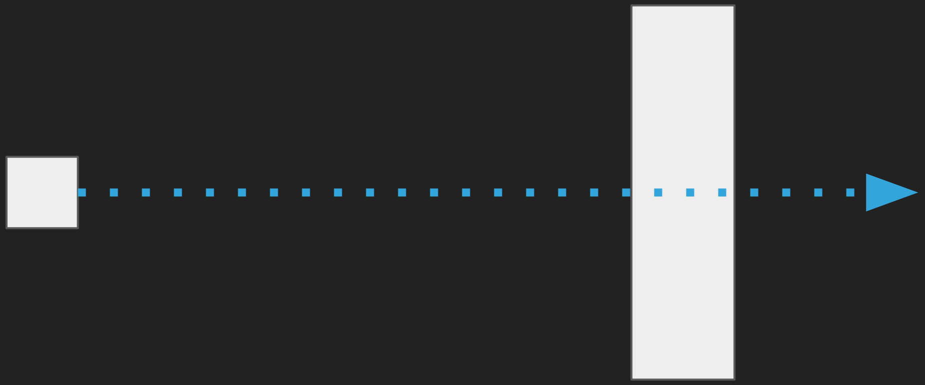
Currently, our timestep is as fast as our computer can go as well as variable.



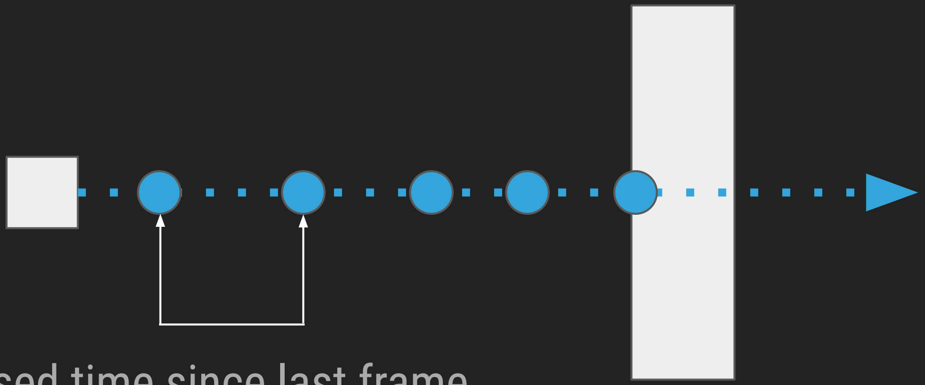
```
float lastTicks = 0;

void Update() {
    float ticks = (float)SDL_GetTicks() / 1000.0f;
    float deltaTime = ticks - lastTicks;
    lastTicks = ticks;

    player_position += player_movement * deltaTime;
}
```

Variable Timestep

Variable Timestep

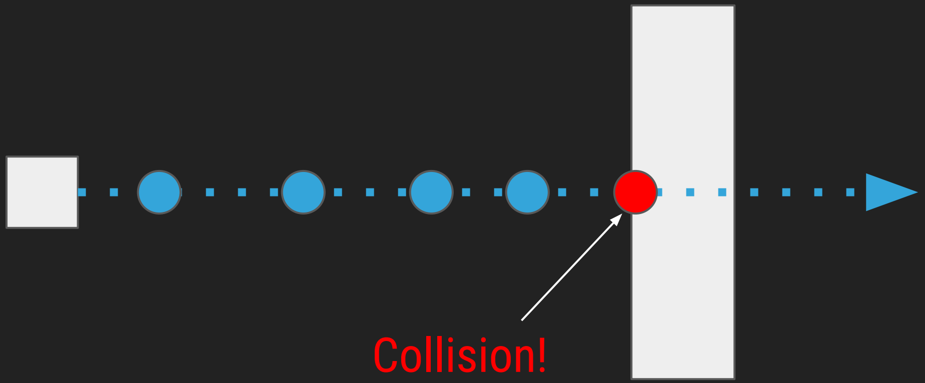


Variable Timestep



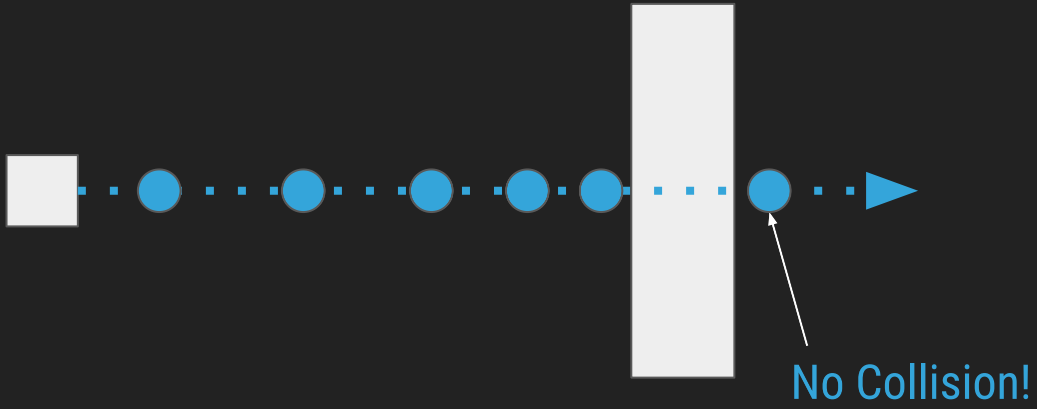
Elapsed time since last frame

Variable Timestep



(everything worked out OK here)

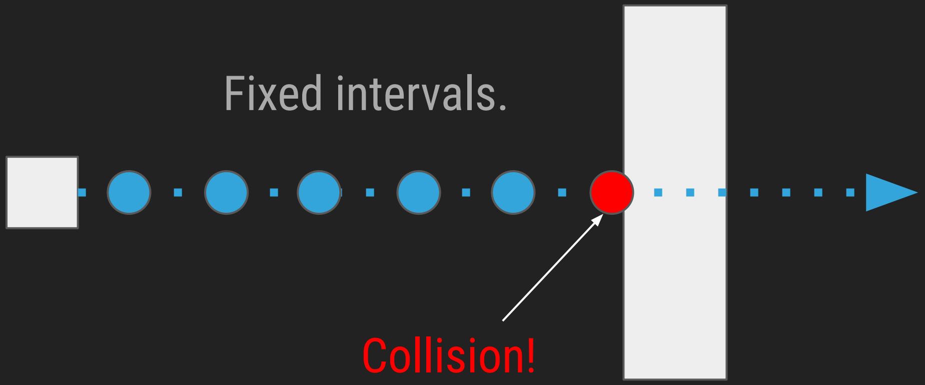
Variable Timestep



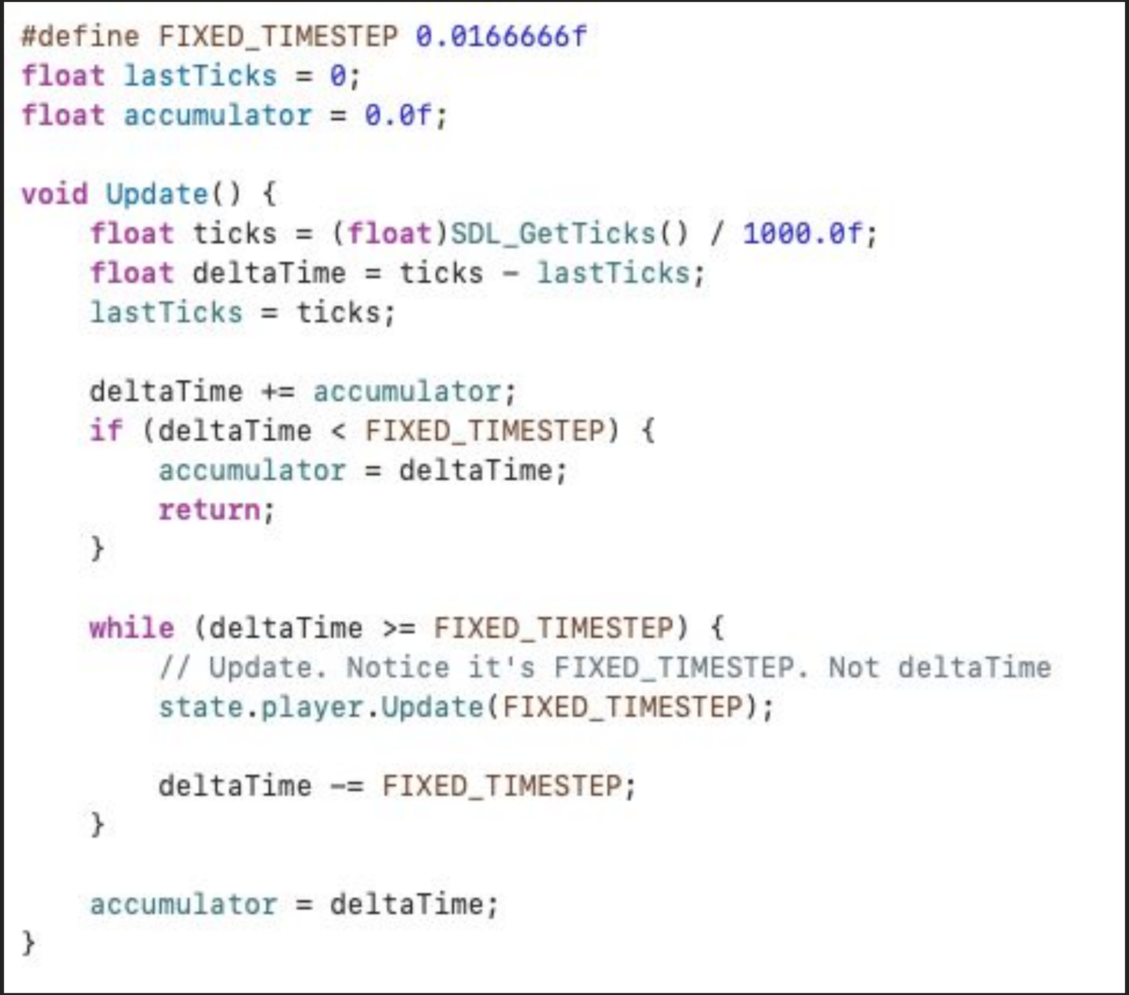
Due to our variable timestep, we “skipped over” the object.

To keep physics behaviors the same, we want to use a fixed timestep.

Fixed Timestep



(everything worked out OK here)



```
#define FIXED_TIMESTEP 0.0166666f
float lastTicks = 0;
float accumulator = 0.0f;

void Update() {
    float ticks = (float)SDL_GetTicks() / 1000.0f;
    float deltaTime = ticks - lastTicks;
    lastTicks = ticks;

    deltaTime += accumulator;
    if (deltaTime < FIXED_TIMESTEP) {
        accumulator = deltaTime;
        return;
    }

    while (deltaTime >= FIXED_TIMESTEP) {
        // Update. Notice it's FIXED_TIMESTEP. Not deltaTime
        state.player.Update(FIXED_TIMESTEP);

        deltaTime -= FIXED_TIMESTEP;
    }

    accumulator = deltaTime;
}
```

Gravity

Gravity

(Acceleration due to Gravity)

9.81 m/s^2

Gravity

(Acceleration due to Gravity)

```
player.acceleration = glm::vec3(0, -9.81f, 0);
```

Acceleration

Rate of change of velocity.

```
velocity.x += acceleration.x * elapsed;
velocity.y += acceleration.y * elapsed;

// You can also do this
velocity += acceleration * elapsed;
```

Velocity

Change of position over time.

```
position.x += velocity.x * elapsed;
position.y += velocity.y * elapsed;

// You can also do this
position += velocity * elapsed;
```

Putting it all together:

```
player.acceleration = glm::vec3(0, -9.81f, 0);

void Update(float deltaTime) {  // player’s update
    velocity += acceleration * deltatime;
    position += velocity * elapsed;
}
```

Notice if acceleration never changes,
velocity will keep accumulating.

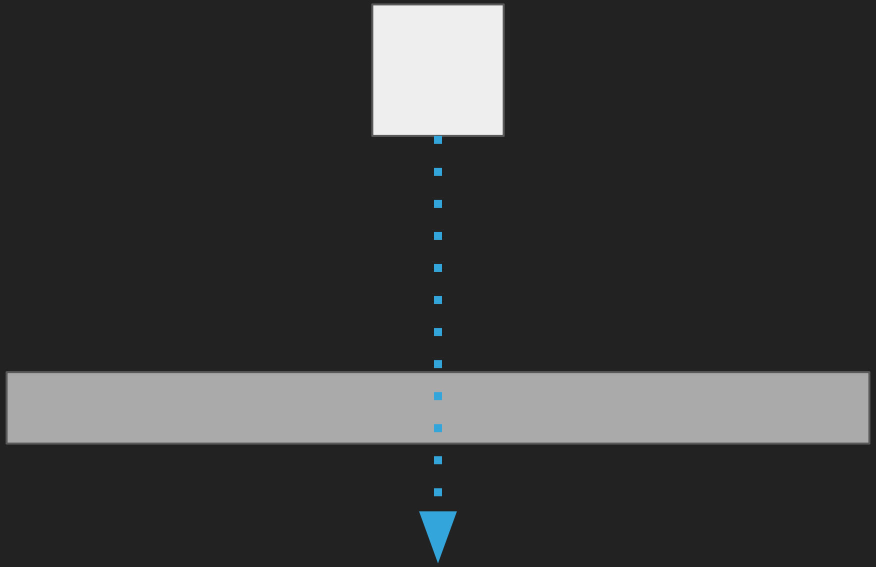
Let’s Code!

Example: Fixed Timestep

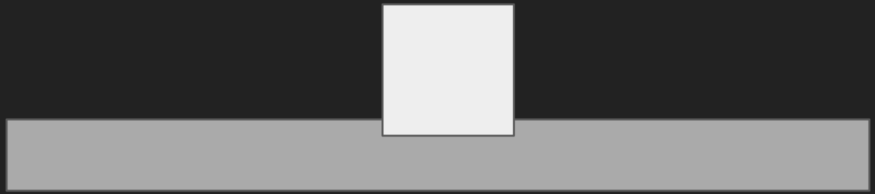
Review
Add platforms
Add collision detection

We Got Stuck!

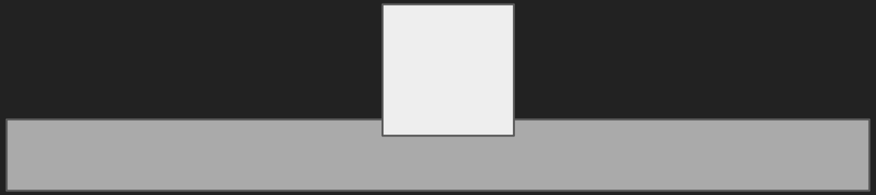
Check for Overlap



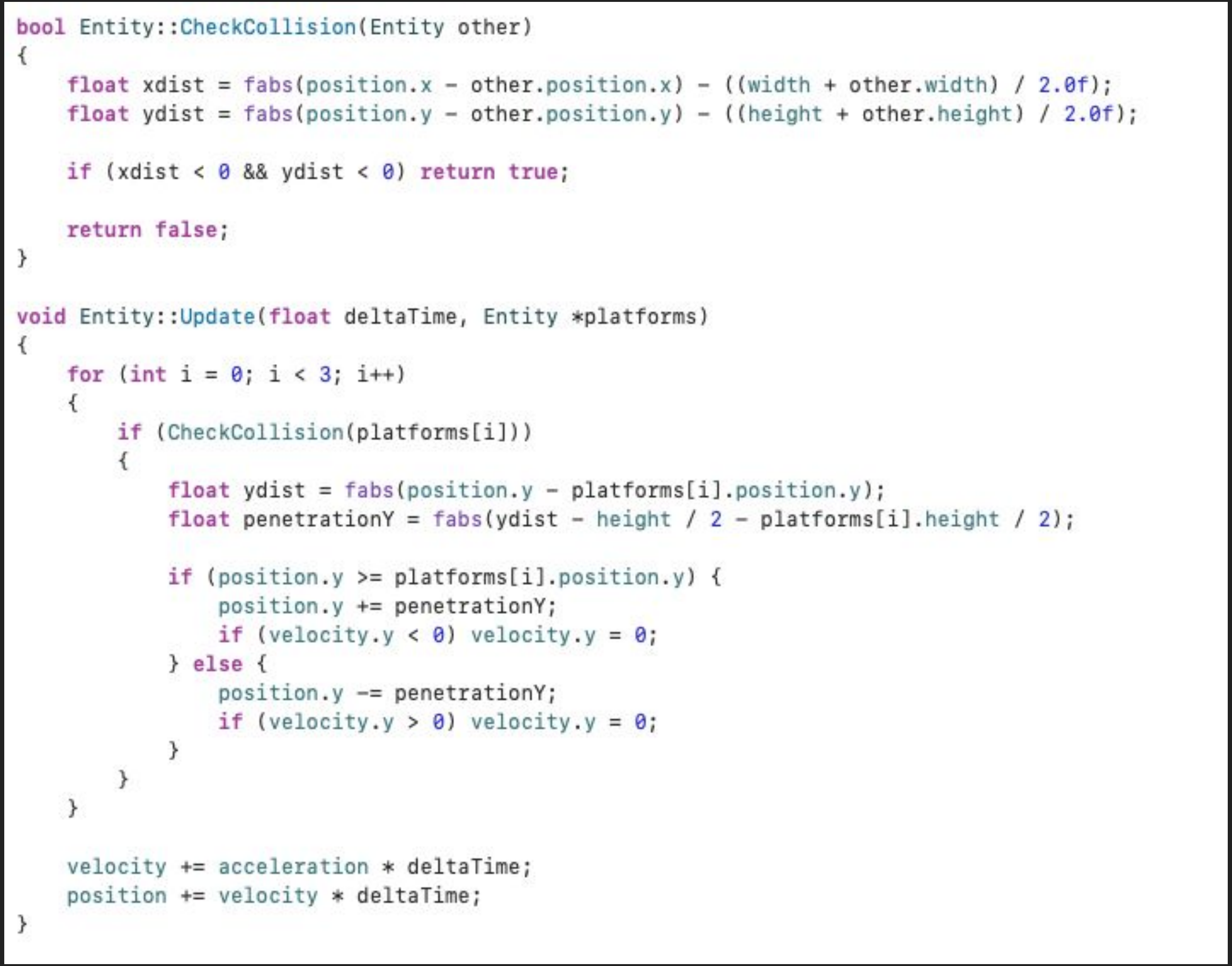
Check for Overlap



Check for Overlap



```
float ydist = fabs(position.y - platform.position.y);
float penetrationY = fabs(ydist - height / 2 - platform.height / 2);
```



```cpp
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) return true;

    return false;
}

void Entity::Update(float deltaTime, Entity *platforms)
{
    for (int i = 0; i < 3; i++)
    {
        if (CheckCollision(platforms[i]))
        {
            float ydist = fabs(position.y - platforms[i].position.y);
            float penetrationY = fabs(ydist - height / 2 - platforms[i].height / 2);

            if (position.y >= platforms[i].position.y) {
                position.y += penetrationY;
                if (velocity.y < 0) velocity.y = 0;
            } else {
                position.y -= penetrationY;
                if (velocity.y > 0) velocity.y = 0;
            }
        }
    }

    velocity += acceleration * deltaTime;
    position += velocity * deltaTime;
}
```

Let’s Code!

Update Collision Detection!

Add Jumping!

Add Moving!