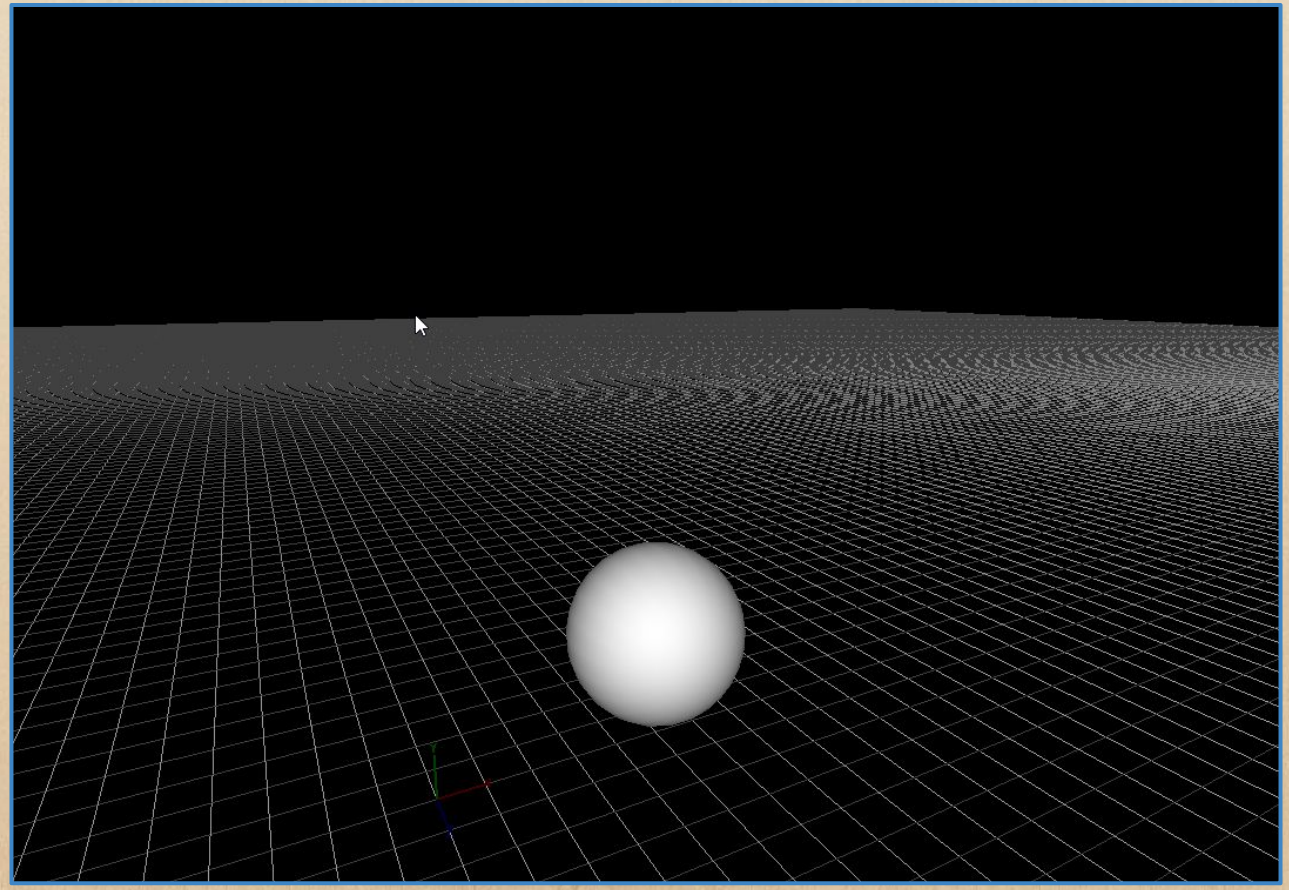

Collision
Detection

Mixing primitives and rigidbodies

It's time to move all shapes according to the physical laws and to detect all collisions between the different primitives.

- X We will use an intermediate class (PhysBody3D) as we did with Box2D.
- X We will look for collisions and give a response (OnCollision method) for each body collided trough all modules as listeners of these collisions.



Our goal

YOUR TURN !

TODO No 1

```
// TODO 1: Store all collision shapes
// TODO 1: Store all bodies
```

Memory leaks! In order to keep an eye on every new allocated memory, we will store them into lists. Save collision shapes and PhysBody3D!

Take into account that collision shapes can be reused whether we are creating physical bodies with the same shape

TODO No 2

```
// TODO 2: Free all collision shapes and bodies
```

Now, you can free all pointers in the CleanUp method.

You have an example of how to delete the collision objects. Think on rigidbodies! Remember that calling Clear() doesn't delete all the data.

TODO No 3

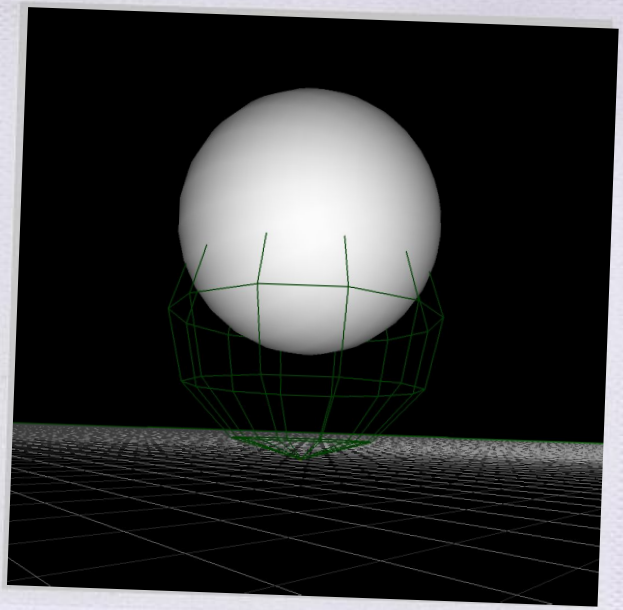
```
// TODO 3: create a sphere in the world with a primitive
// and create a physics body for it
```

Create a primitive Sphere and set its position above the ground, and recall to render the sphere on the Update().

Afterwards, create a rigidbody for that sphere and the ball will fall when starting the application.

TODO No 3

BUT! We have to synchronize the position of our white primitive sphere with the position of our physical sphere. We need to go the next TO DO.



TODO No 4

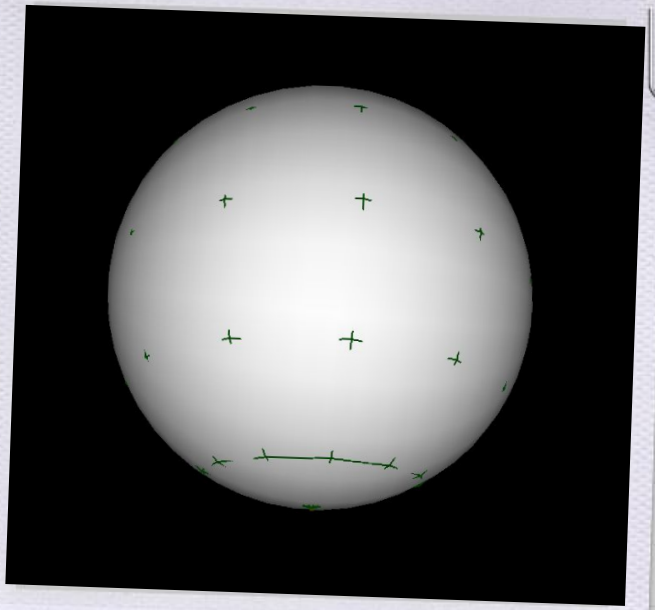
```
// TODO 4: update the transform of the shape
to meet the physics one
// You have to fill in PhysBody methods.
Remember to use get/setOpenGLMatrix methods
```

Fill in the different methods of PhysBody3D class. The idea is to work with **btTransform** and set/get this transform from the matrix values.

Once completed, you can update the sphere position with the Bullet representation.

TODO No 4

Now, both entities are synchronized!



TODO No 5

```
// TODO 5: Add this module to the list of collision listeners
// ... and define it for the ModuleSceneIntro. Set the ball
// in red if it happens using it's color property
```

Similarly to Box2D, each PhysBody3D has a set of Module listeners that will give response to each collision. ModuleSceneIntro will be a listener for the sphere.

Change the white ball to **red** when collides!

TODO No 6

```
// TODO 6: Detect collisions:
// - Iterate all manifolds
// - Count the number of contacts
// - If we have contacts, get both PhysBody3D from userpointers
// - If iterate all contact listeners and call them
```

First, use *world->GetDispacher()* to get number of manifolds. Then, you can obtain each specific manifold with *getManifoldByIndexInternal()*. For each *btPersistentManifold** you can check for the number of contacts and find out the bodies involved in the collision. Finally, use *getUserPointer()* from each body to get the PhysBody3D* and iterate all collision listeners to call OnCollision for each of them.

Homework

Create methods to spawn planes, boxes, spheres and cylinders with their corresponding physics bodies.

Extra: When pressing '1', throw spheres in the direction that camera is looking at.

Next Week . . .

Bullet Constraints