Physics 2 -Collision Detection

Ricard Pillosu - UPC

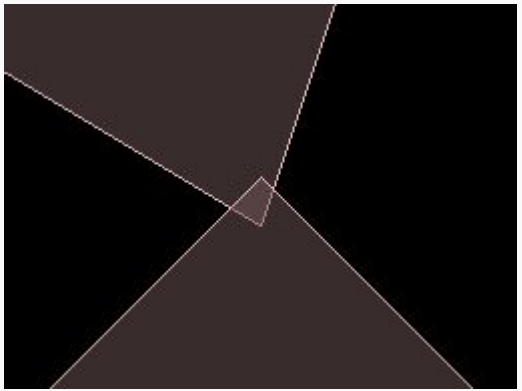
Collision Module

- It manages the creation of shapes, collision tests and binary functions:
- **Shapes**: circles, polygons and chains
- **Binary functions**: Overlap, Contact, Distance, Time of Impact
- **Tests**: point, raycast

Overlap

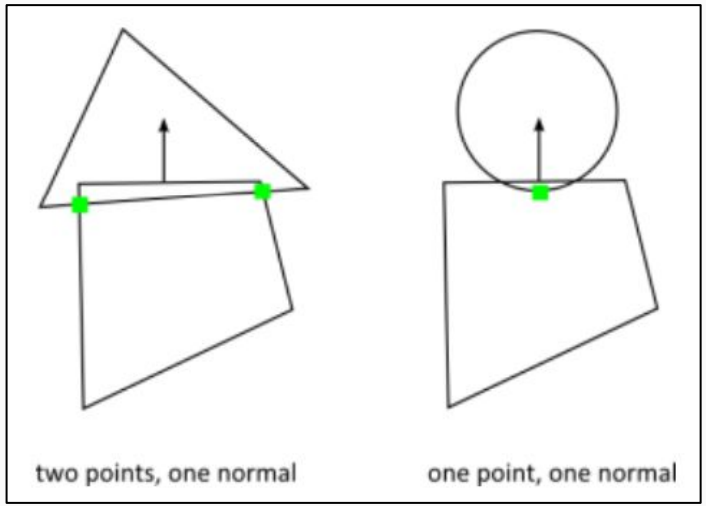
- Just tells if two shapes overlap in any point:

```
bool overlap = b2TestOverlap(
     shapeA, indexA,
     shapeB, indexB,
     transformA,
     transformB);
```



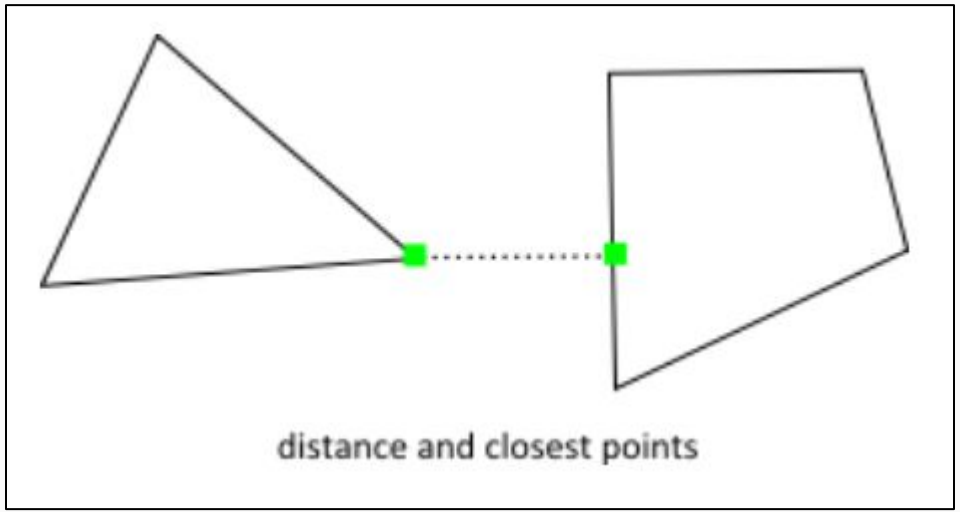
Contact

- Given two shapes, returns point in space of contact and normals:



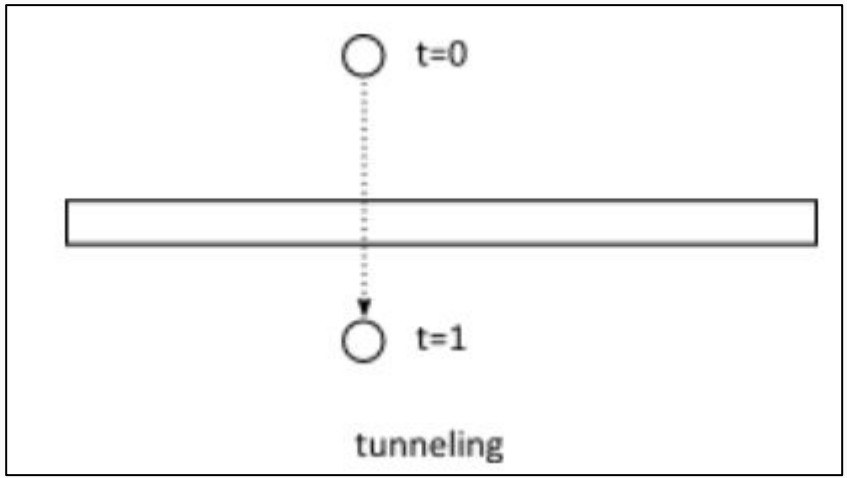
Distance

- Given two shapes, returns distance and closes points (**b2Distance**):



Time of Impact

- Given two shapes, **b2TimeOfImpact** returns seconds (in simulation) to collision:



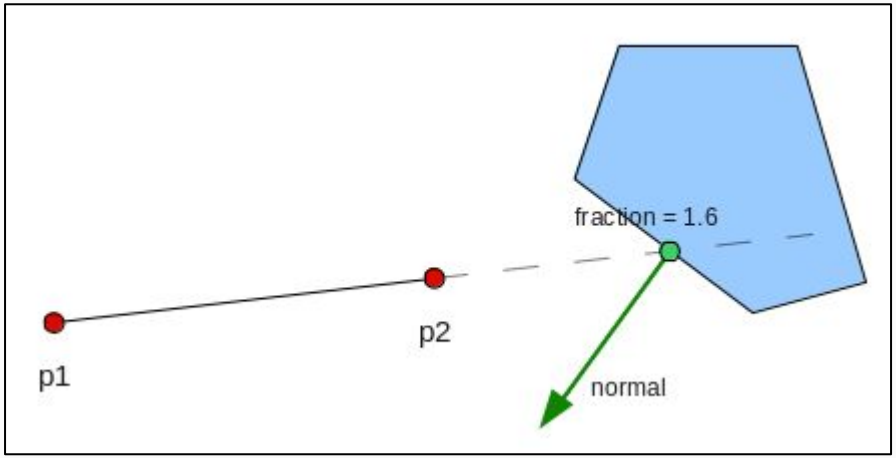
TODO 1

“Write the code to return true in case the point is inside ANY of the shapes contained by this body”

- We can test if a point is inside a shape
- It is a method inside **shape** (body -> many fixtures -> 1 shape)
- Needs to know it’s current transformation (body->GetTransformation())
- Remember to always transform from pixels to meters!

Raycasting

- Method per shape, it needs the transformation too
- Gets two points and fraction
- Returns fraction and normal



TODO 2

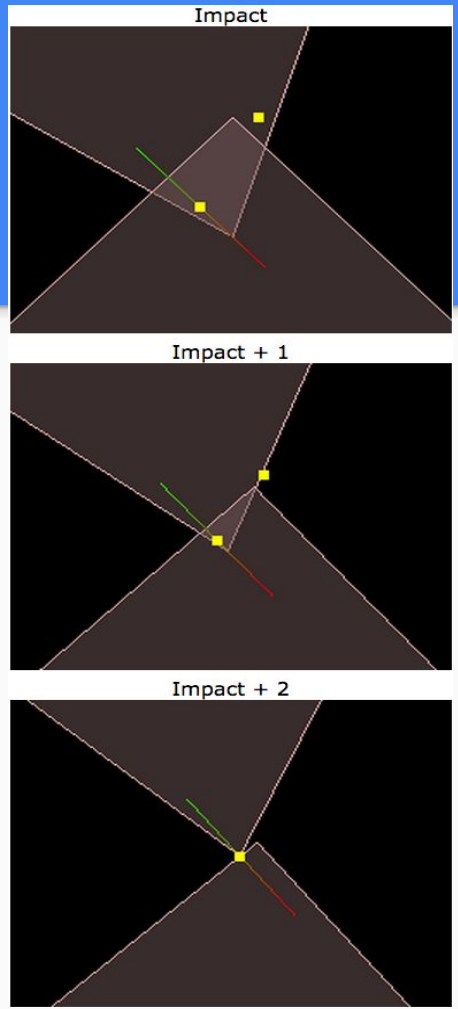
“Write code to test a ray cast between both points provided. If not hit return -1. If hit, fill normal_x and normal_y and return the distance between x1, y1 and its colliding point”

- Code is similar to previous TODO
- Use **output.fraction** to calculate the returning value
- Yes, this needs some maths from you!

Collision Detection

- Box2D reacts to overlapping shapes pushing them away
- It does in [different steps]:
  - BeginContact
  - PreSolve
  - PostSolve
  - EndContact

One per step!



TODO 3

“Make module physics inherit from b2ContactListener then override void BeginContact(b2Contact contact)”*

- You need to make *ModulePhysics* class a contact listener
  - *world->SetContactListener(b2ContactListener*)*
- Just LOG “collision!” and check if it works

TODO 4

“Add a pointer to PhysBody as UserData to the body”

- Bodies have a void pointer to user data for our convenience
- Do it in all methods that create a body
- We can use it to store any pointer!
- Use it to store the address of our *PhysBody* class
- Now inside collision callback, obtain pointers to both *PhysBody**

TODO 5

“Create a OnCollision method that receives both PhysBodies”

- We will specialize this method to modules interested in this
- It should receive both *PhysBodies* involved in the collision
- Avoid includes, use forward declarations

TODO 6

“Add a pointer to a module that might want to listen to a collision from this body.”

- *PhysBodies* should know which module (if any) want to listen to collisions
- Make sure the pointer is NULL when the class is constructed

TODO 8

“Now just define collision callback for the circle and play bonus_fx audio”

- Make sure to add *ModuleSceneIntro* class as listener to all circles
- Define the collision callback and just play bonus_fx sound

Homework

- Find out about sensors in Box2D
- Sensors won’t use callbacks for collision detection
- You need to iterate all contacts and keep those that collide ***IsTouching***()
- Then call collision callback if same as any other body
- I suggest you use *PreUpdate* method for this, just after stepping the world.