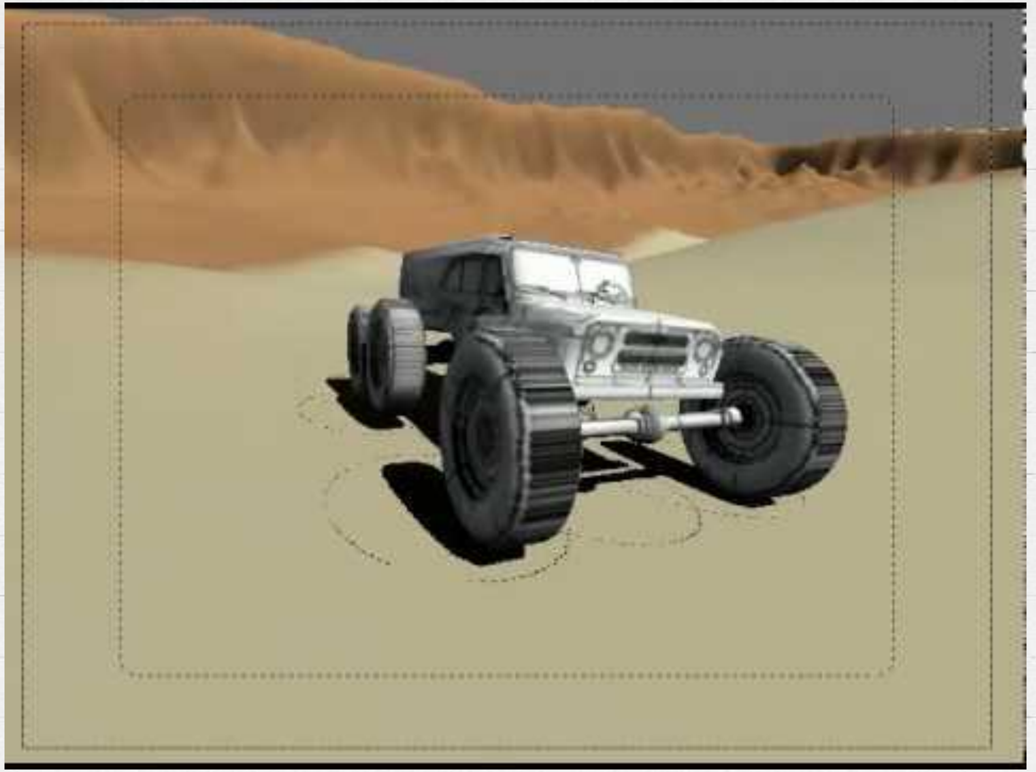
Vehicle Creation

A vehicle

With everything that we've learnt, how would you do a car?



A vehicle

RigidBodies, Constraints with motors, Springs...

That sounds like a lot of work.

btRayCastVehicle

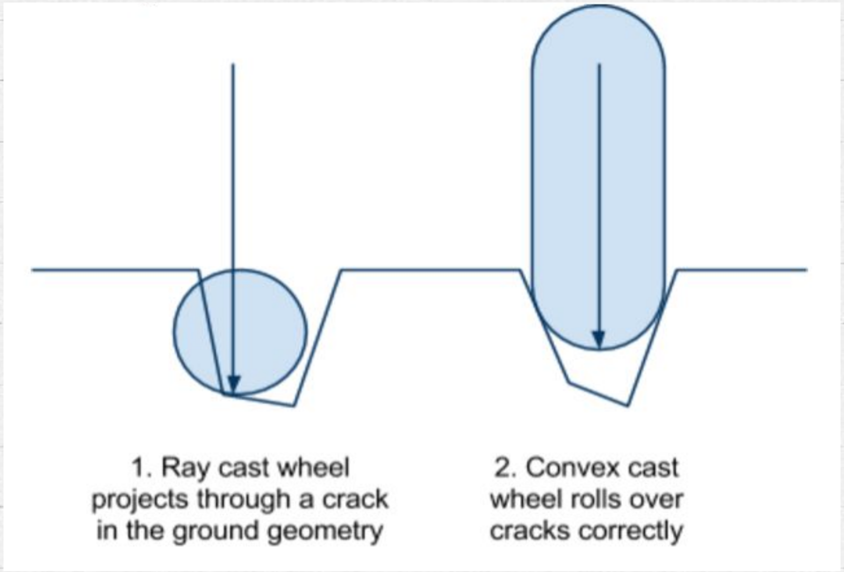
Kester Maddock created a driving model system!

Meet the

btRayCastVehicle

btRayCastVehicle

"The entire vehicle is represented as a single rigidbody, the chassis. The collision detection of the wheels is approximated by ray casts, and the tire friction is a basic anisotropic friction model."


1. Ray cast wheel projects through a crack in the ground geometry
2. Convex cast wheel rolls over cracks correctly

Vehicle info struct

```
struct VehicleInfo
{
    ~VehicleInfo();

    vec3 chassis_size;
    vec3 chassis_offset;

    float mass;
    float suspensionStiffness; // default to 5.88 / 10.0 offroad / 50.0 sports car / 200.0 F1 car
    float suspensionCompression; // default to 0.83
    float suspensionDamping; // default to 0.88 / 0..1 0 bounces / 1 rigid / recommended 0.1...0.3
    float maxSuspensionTravelCm; // default to 500 cm suspension can be compressed
    float frictionSlip; // defaults to 10.5 / friction with the ground. 0.8 should be good but high values feels
better (kart 1000.0)
    float maxSuspensionForce; // defaults to 6000 / max force to the chassis

    Wheel* wheels;
    int num_wheels;
};
```

Wheel struct

```
struct Wheel
{
    vec3 connection; // origin of the ray. Must come from within the chassis
    vec3 direction;
    vec3 axis;
    float suspensionRestLength; // max length for suspension in meters
    float radius;
    float width;
    bool front; // is front wheel ?
    bool drive; // does this wheel received engine power ?
    bool brake; // does breakes affect this wheel ?
    bool steering; // does this wheel turns ?
};
```

PhysVehicle struct

```
struct PhysVehicle3D : public PhysBody3D
{
public:
    PhysVehicle3D(btRigidBody* body, btRaycastVehicle* vehicle, const VehicleInfo& info);
    ~PhysVehicle3D();

    void Render();
    void ApplyEngineForce(float force);
    void Brake(float force);
    void Turn(float degrees);
    float GetKmh() const;
public:

    VehicleInfo info;
    btRaycastVehicle* vehicle;
};
```

Vehicle Creation

To create a vehicle, we need to fill in and tweak the different parameters in the *VehicleInfo* struct.

Then, we just call App->physics->AddVehicle(const &VehicleInfo car);

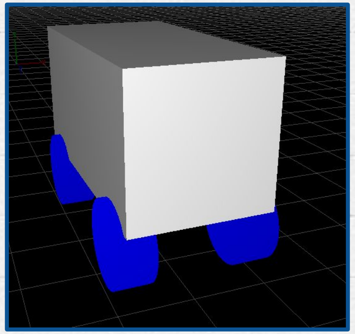
We can control the car in an Update loop! (Create PlayerModule?)

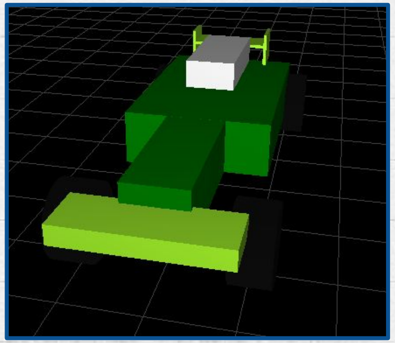
Vehicle Creation

Once you understand the code, try to go crazy:

- Change the chassis of your car:

  A Formula 1, a jeep, an articulated bus, a van, a forklift?





Improve the car!

Once you understand the code, try to go crazy:

- X Tweak the values: it depends on the type of vehicle that you want

Improve the car!

Once you understand the code, try to go crazy:

X How many wheels do you need?

The camera

The camera can make or break your game. Give it some love.

It's going to be harder than it looks. We're in 3D now.

How will it behave in your game? A static/dynamic panoramic view, a chasing camera...? Controller with the mouse, keyboard?

The LookAt method on camera will be very useful for this assignment.

Homework

We now have everything we need to make our game (or almost).

Create the circuit, try and find a place to use constraints.

Create the camera for your game.

Remember we're making a game! Which is the objective? How do we win? How do we lose?

Homework

Brainstorm! Try different things.

How is the car?

How many laps?

Obstacles? What can we do with constraints?

Landmines?

Drifting?

More than a single car?

Homework

Next class, try to have a "playable" version of the game.

Having other people try it can be a BIG help!

Aim to have an "almost finished" game. Polishing it to make it fun takes longer than you'd think.

Next Week . . .

Sensors