orxBODY structure

Summary

Body

```
[BodyTemplate]
PartList           = BodyPartTemplate1#BodyPartTemplate2#...
AllowGroundSliding = <bool>
AngularDamping     = <float>
CustomGravity      = <vector>
Dynamic            = <bool>
FixedRotation      = <bool>
HighSpeed          = <bool>
Inertia            = <float>
LinearDamping      = <float>
Mass               = <float>
```

BodyPart

Sphere

```
[BodyPartSphereTemplate]
Type        = sphere
Center      = <vector> | full
Radius      = <float> | full
CheckMask   = <16b flags>
SelfFlags   = <16b flags>
Density     = <float>
Friction    = <float>
Restitution = <float>
Solid       = <bool>
```

Box

```
[BodyPartBoxTemplate]
Type        = box
TopLeft     = <vector> | full
BottomRight = <vector> | full
CheckMask   = <16b flags>
SelfFlags   = <16b flags>
Density     = <float>
Friction    = <float>
Restitution = <float>
```

```
Solid       = <bool>
```

Mesh (polygon)

```
[BodyPartMeshTemplate]
Type        = mesh
VertexList  = <vector>#<vector>#...
CheckMask   = <16b flags>
SelfFlags   = <16b flags>
Density     = <float>
Friction    = <float>
Restitution = <float>
Solid       = <bool>
```

Details

Body

Here's a list of the available properties for an `orxBODY` structure:

- `PartList`: List of all the parts that will compose a body. There's no limit on the number of parts that can be defined for a single body. This property *needs* to be defined if you want it to collide with other bodies.
- `AllowGroundSliding`: If set to false on a dynamic object, it will prevent it from sliding on static slopes of more than 45°. This only works with a vertical top-down gravity. By default its value is `true`.
- `AngularDamping`: Damping of angular velocity for this body. By default its value is 0.0, which means no damping.
- `CustomGravity`: Defines a gravity vector to use for this body instead of the world's one. By default it doesn't exist, which means world's gravity will be used for this body.
- `Dynamic`: Defines if this body should be dynamic or static. If your object is expected to move, this property should be set to `true`. Static bodies can't collide with other static bodies. By default, its value is `false` (ie. static).
- `FixedRotation`: Defines if your dynamic object is allowed to rotate as a result of collision forces. By default its value is `false` which means it can rotate freely.
- `HighSpeed`: For high velocity objects (like bullets), this property should be set to `true` to avoid collision errors. However, every object flagged as `HighSpeed` will cost more when processed by the physics engine. By default its value is `false`.
- `Inertia`: Defines an inertia value for this body. By default its value is 0.0.
- `LinearDamping`: Damping of speed (linear velocity) for this body. By default its value is 0.0, which means no damping.
- `Mass`: Defines a mass, in kg, for this body. If parts are defined, the mass will be overridden by an automatically calculated value based of parts' sizes and positions.

BodyPart

Common

Here's a list of the available properties for all types of body parts:

- `Type`: Defines the type of the body part. Available types are `sphere`, `box` and `mesh` (ie. convex polygon). This property *needs* to be defined.
- `CheckMask/SelfFlags`: Both properties are flags expressed on 16bits. The `SelfFlags` defines this part identity whereas the `CheckMask` defines which parts are allowed to collide with it. For a collision to happen between two parts the expressions (Part1.CheckMask & Part2.SelfFlags) and (Part2.CheckMask & Part1.SelfFlags) have both to evaluate to `true`. NB: Two parts of the same body won't collide whichever `CheckMask/SelfFlags` they have. [1)]
- `Density`: Defines the density of this part. Its default value is 0.0, which means it doesn't have any influence on the body's mass.
- `Friction/Restitution`: Define the friction and restitution of this part, usually between 0.0 and 1.0. [2)] By default both their values are 0.0.
- `Solid`: Defines if this part is solid or not. Only solid parts will trigger a reaction on their body when colliding with others. By default its value is `false` which means the collision info will be signaled through events, but the physics simulation of this body won't be automatically affected by it.

Sphere

Here's a list of the available properties only available to sphere parts:

- `Center`: Defines the center of the sphere (in 2D it's a circle, of course) in the parent's space (ie. in object's space). By default its value is `full` which means the center will match the object's one (ie. the center of its current graphic).
- `Radius`: Defines the radius of the sphere (or 2D circle). By default its value is `full` which means the sphere's radius will match the biggest dimension of the parent object. You can find an example in the spawner tutorial [3)].

Box

Here's a list of the available properties only available to box parts:

- `TopLeft/BottomRight`: Define the extrema of the box (in 2D it's a rectangle, of course) in the parent's space (ie. in object's space). By default their values are `full` which means `TopLeft` and `BottomRight` will match the full rectangle defined by the parent object's current graphic. You can find an example in the physics tutorial.

Mesh (polygon)

Here's a list of the available properties only available to mesh [4)] parts:

- `VertexList`: Provides a list of vertex coordinates in parent object's space. The resulting

polygon *needs* to be convex. Up to 8 vertices can be defined and they ***have to be entered clockwise***. You can find an example in the spawner tutorial [5].

[1] Check the documentation of Box2D for more information on filtering

[2] Check the documentation of Box2D for more information on friction/restitution

[3], [5] by looking directly at the config files as they're not covered in the wiki

[4] convex polygon