Hex Engine

Shawn Deprey : Blackmodule Studio

Hex Engine in an HTML5 game engine by Shawn Deprey developed using Javascript and the HTML5 Canvas object. The engine was developed to be easy to use and is great for game development beginners with programming experience.

Physics

Physics.js

The Hex physics engine was developed to be modular and easy to use. The engine uses earths gravitational and mass constants which makes the gravity simulation realistic. These constants can be modified in the constructor of **Physics.js**. There are 4 functions and 4 properties that allow an engine to use Hex physics: **addEntity(Entity), addCollisionArea(x, y, height, width), applyGravity(), calculateCollisions(), YourEntity.height, YourEntity.width, YourEntity.x, YourEntity.y**

Hex physics also adds properties to your entities when you pass them into the engine. Those properties are: velocity, **stepSize, collideTop, collideBottom, collideLeft, collideRight, grounded, collision\_id**. These properties can be accessed for your own game logic.

float(stepSize) boolean(collideTop, collideBottom, collideLeft, collideRight, grounded) int(collision\_id)

You can pass in any entity you with to apply physics; the only requirement is that the entity must have **height, width, x and y** properties. There is also an optional **stepSize** property which is a percent you want your entities to be able to “step up” onto a collision area.

**Example**

function Entity\_Player()

{

this.height = 128;

this.width = 128;

this.x = 0;

this.y = 0;

this.stepSize = 0.25; //% between 0 - 1

}

function World()

{

var physics = new Physics();

var player = new Entity\_Player();

this.init = function()

{

physics.addEntity(player);

physics.addCollisionArea(450, 290, 15, 900);

physics.addCollisionArea(867, 302, 10, 100);

}

this.update = function()

{

player.update();

physics.applyGravity();

physics.calculateCollisions();

}

}