Project Info

Status

Well, the samples are running at a reasonable frame rate, so I'd call it 'beta'.

Currently there is no road map, so the future of this project is also up to the community, i.e. *you* ;-)

Overview

ArcadeJS is a 2d game engine, written in pure JavaScript. It requires HTML5, namely support for canvas and audio elements.

This package was developed during a summer hollyday 2010 and finished during winter 2010. The goal of this fun project was to learn about HTML5 and implement a clone of the [Rip-Off](#) game. As a consequence I borrowed a lot (especially processing.js), reinvented some wheels and stuck with rather simple vector graphics for the demo games.

Since it is HTML5, the demos run on smart phones as well.

Part of the distribution is

- lina.js
  An independent object oriented math library for points, vectors, and homogeneous transformations in 2D space.
  A polygon class helps with collision detection and hit testing.
  See [LinaIntro](#) for details.

- arcade.js
  A 2D game engine that drives a render loop for multiple moving objects.

Also sound, keyboard, mouse and touch events are supported.
See [ArcadeIntro](#) for details.

- Arcade-controls.js
  Some controls (button, joystick) for mouse and touch screens.

- Some demos
  See http://arcade-js.googlecode.com/hg/src/demos/index.html

ArcadeIntro

A 2D game engine that drives a render loop for multiple moving objects.
Also sound, keyboard, mouse and touch events are supported.

The linear algebra math is provided by LinaJS, so you might want to read [LinaIntro](#) first.

Details:

> http://docs.arcade-js.googlecode.com/hg/arcade.js/jsdoc/symbols/ArcadeJS.html

LinaIntro

An independent object oriented math library for points, vectors, and homogeneous transformations in 2D space.

Details:

> http://docs.arcade-js.googlecode.com/hg/lina.js/jsdoc/symbols/LinaJS.html

TODO: sample

Note pitfall

> Transformations and most other methods operate 'in-place', thus modifying the object itself.

> Don't forget to .copy()

ArcadeJS Tutorial

Table of Content

Overview

The open source project home is located at http://arcade-js.googlecode.com.

TODO: COPY OF ArcadeIntro

Creating a game page

To implement a game, we need

1. One HTML page that includes the required JavaScript libraries and one canvas element. The Arcade-JS game object is instantiated here.

2. The game code

```
<!DOCTYPE html>
<html>
<head>
  <script src="../../depends/jquery.js"></script>
  <script src="../../lina.js/lina.js"></script>
  <script src="../../arcade.js/arcade.js"></script>
  <script src="jsAsteroids.js"></script>

  <script type="text/javascript">
  $(function(){
    // Initialize the game
    var canvas = document.getElementById("gameCanvas");
    var game = new AsteroidsGame(canvas);
  });
  </script>
</head>
<body>
<canvas id="gameCanvas">This game requires HTML 5 support.</canvas>
</body>
</html>
```

Render loop

A game is essentially an infinite sequence of scene snaphots (or 'frames').

```
while game.isRunning:
   // Set up the new scene positions
   for obj in game.object_list:
     calculate_new_object_position(obj)
     // Let object modify this
     obj.step()

   // Draw all objects
   clear_canvas()
   for obj in game.object_list:
     set_canvas_context(obj.pos, obj.orientation)
     // Let object draw itself, using modelling coordinates
     obj.draw()
```

More detailed:

```
while game.isRunning:
   // Trigger timeout event
   if game.timeout_reached:
     game.onTimeout()

   // --- Step all objects
   game.preStep()

   for obj in game.object_list:
     // Trigger timeout event
     if obj.timeout_reached:
          obj.onTimeout()

     // Calculate new position
     obj.pos += obj.velocity
     obj.orientation += obj.rotationalSpeed
          obj.step()
     if obj.auto_wrap:
          <calculate wrapped position>

     // Let object modify this
     obj.step()

   game.postStep()

   // --- Draw all objects
   clear_canvas()
   game.preDraw(ctx)

   for obj in game.object_list:
     save_canvas_context()
     set_canvas_context(obj.pos, obj.orientation)

     // Let object draw itself, using it's own modelling coordinates
     obj.draw(ctx)

     draw_object_debug_infos()
     restore_canvas_context()

   game.postDraw(ctx)

   draw_game_debug_infos()
```

Frames, velocities and timing

Object velocities (object.velocity and object.rotationalSpeed) are defined in World Coodinate units per second.
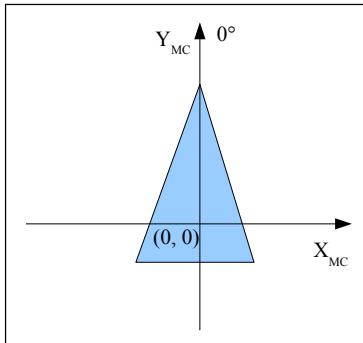
Time correction

TODO

Coordinate systems and transformation pipeline

Modelling Coordinates ('MC')

All game objects are designed in Modelling Coordinates.

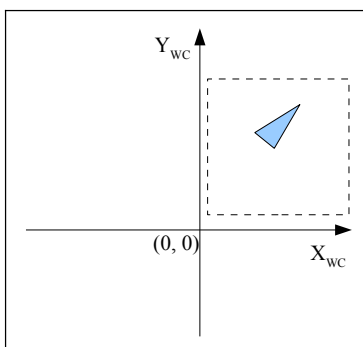The object's neutral orientation is assumed to be upward (along the positive y-axis).
The rotation pivot should be at (0, 0).

World Coodinates ('WC')

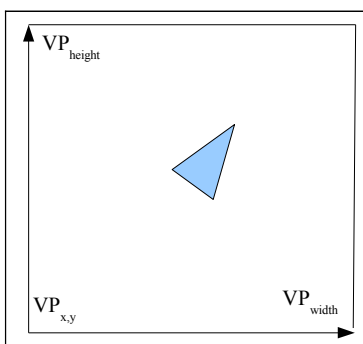The game play takes place in World Coordinates with infinite dimension.
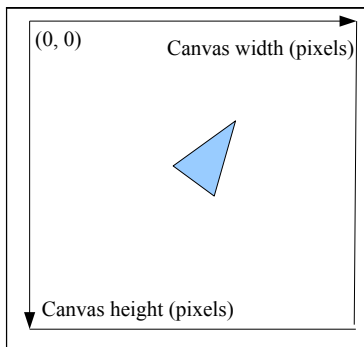The dashed rectangle marks the part of the world is visible to the user ('Viewport').

Viewport

The Viewport defines the visible part of the 'world'.

It's dimensions are specified in World Cordinate units.

Canvas Coordionates ('CC')

Finally the objects are rendered to the canvas using pixel coordinates.
Note that the positve y-axis of the canvas points downward.

Viewport definition

The viewport is defined using World Coordinates

```
game.setViewport(0, 0, 640, 480, "extend");
```

If the aspect ratio of the viewport and the canvas are different, some parts outside of the original viewport may be displayed in order to prevent stretching. The mapMode defines controls this:

- 'stretch'
- 'fit'
- 'extend'
- 'trim'
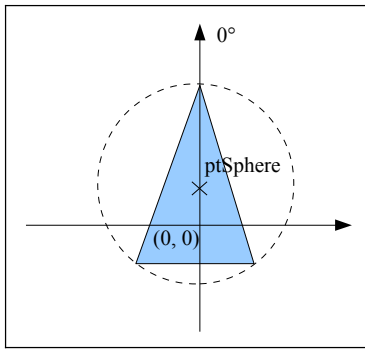- 'none'

Drawing

Augmented canvas

Details:

http://docs.arcade-js.googlecode.com/hg/arcade.js/jsdoc/symbols/ArcadeCanvas.html

Game play

Set up the object list

Collision detection

The center of the bounding sphere is not necessarily identical with the rotation pivot.

Scheduled events (timeout trigger)

- game.later()
- object.later()

Activities

- onActivity(), setActivity(), isActivity()
- ==> API Doc

The object list

Sound

See

http://docs.arcade-js.googlecode.com/hg/arcade.js/jsdoc/symbols/AudioJS.html

User input

Event handling

Keyboard input

Mouse input, Drag'n'drop

Touch events

Controls

Mobile devices and touch events

Debugging

Game.debug

Game.opts.debug.showVelocities = true

stoprequest 0 true

logToCanvas

Further information

The LinaJS API is documented at http://docs.arcade-js.googlecode.com/hg/lina.js/jsdoc/index.html.