CMPS 20 – Game Design Experience

Winter 2013

Week 3: Offline editing, Snake game, project design

Arnav Jhala

TA: Sarah Harmon

Graders: Eric Lipshutz and Andrew Duensing, Thomas Deeb
Guest Speaker

• Teale Fristoe
Updates/Announcements

• Grading update
  – Team formation graded (will be updated after class today)
  – Alchemy (almost done – will update by Sunday)
  – Concept Document (upload comments to eCommons by Wednesday)

• Assignment II
  – Last individual assignment
  – Snake game. Instructions updated on class website.
  – DUE next FRIDAY at 9 PM on eCommons
Project Deliverable

• Game Concept Document
  – A compelling document that sells your game concept
  – Title page
    • Title of game, name of group, name of team members, sample artwork
  – Overview page
    • Table at top: game genre, platform, team size
    • Key points section
      – Bulleted list of important elements of gameplay
      – Goal of game, what makes game unique, main characters, main fictional elements
      – Sample artwork image to give feel of the game
  • How your game addresses class theme
  – Biographies
    • True, pocket biographies of each team member (1-2 paragraphs each) stressing experience that makes you a strong game designer/programmer/artist
  – 1-3 pages giving a textual description of the game
    • Fictional background, brief description of characters, goal of player in game, how does player interact with the game, brief description of levels, game audience, other important elements as needed.
  – 1-2 pages of sample conceptual artwork
    • Hand-drawn sketches are fine
• See template and evaluation criteria on course website
Assignment II

• Aptana Studio
  – Offline editing
  – Managing multiple files
  – Snake game revisited
  – Sample code and Aptana setup
Upcoming classes

• Discuss design feedback
• Labs next week
  – Aptana Studio
  – Code arrangement for Snake with multiple files
• Particle Systems
• Project
  – Upcoming documents
  – Mentor assignments
Particle Systems

- A particle system is a technique for modeling “fuzzy” things
  - Used to simulate explosions, fire, smoke, flowing water, sparks, fog, snow, and others
  - A large number of small moving particles combine together to create the effect

www.cgtutorials.com
• **Particle Entity**: Simple, contains information about position, mass, speed, acceleration, age...

• **Particle system**: Contains a group of particles and rules of how they will behave.

• **Emitter**: Way of creating the particles

• **Integrator**: Particles “Update()”

• **Particle Engine**: Group of N particle systems
Parameters of a Particle System

- **Emitter**
  - location in space that acts as the source of the particles
- **Spawn rate**
  - How many particles generated per unit time
- **Initial velocity vector**
  - Direction and speed of particles when created
- **Particle lifetime**
  - How long the particles last
- **Particle color**
  - Color of each particle
  - Can also texture map
- Often values are specified as a center value, with allowable variation
  - Lifetime is 60 ticks +/-20 ticks
Object-oriented programming in JavaScript

• Concepts
  – Classes/Prototypes
  – Encapsulation
  – Inheritance
  – Polymorphism
  – Constructors
  – *new* keyword
• Objects in Javascript
• Properties of objects
• Every object has a *prototype* property
  – It contains reference to an object

```javascript
function Person(gender) {
  this.gender = gender;
  alert('Person instantiated');
}

var person1 = new Person('Male');
var person2 = new Person('Female');

//display the person1 gender
alert('person1 is a ' + person1.gender); // person1 is a Male
function Person(gender) {
    this.gender = gender;
    alert('Person instantiated');
}

var person1 = new Person('Male');
var person2 = new Person('Female');

//display the person1 gender
alert('person1 is a ' + person1.gender); // person1 is a Male

function Person(gender) { 
    this.gender = gender;
}

Person.prototype.sayGender = function()
{
    alert(this.gender);
};

var person1 = new Person('Male');
var genderTeller = person1.sayGender;

person1.sayGender(); // alerts 'Male'
genderTeller(); // alerts undefined
alert(genderTeller === person1.sayGender); // alerts true
alert(genderTeller === Person.prototype.sayGender); // alerts true
function Person(gender, name) {
    this.name = name;
    this.gender = gender;
}
Person.prototype.name = 'foo';

var person3 = new Person('Female', 'Priesha');
person3.name;
>>> "Priesha"

delete person3.name;
>>> true

person3.name;
>>> "foo"
var a = [1,2,3];
for (var i in a){
    console.log(a[i]);
}

var obj = {p1: 1, p2: 2};
for(var i in o){
    for(var i in o){
        console.log(i + ‘=‘ + o[i]);
    }
}
Some interesting properties

hasOwnProperty(), isPrototypeOf()

person1.hasOwnProperty('gender');
>>> true;

Person1.hasOwnProperty('name');
>>> false;