Apollolumi Design Specifications

Use this as a guide when developing assets for Apollolumi!

< TODO – detailed game mechanics/description here >

(Sue me, I’m sick.)

Scroll down to the section that applies most to you…

General Game Flow

Start 🡪 Menu 🡪 Intro 🡪 (Round 🡨🡪 Cutscene) 🡪 Outro 🡪 Back to Menu

Last updated 9-7-13.

Programming

Primary Goal: Create various *enemies* and *turrets*.

Process: 1. Claim or come up with an idea (idea list under “What to do?”)

2. Mail Alex for confirmation

3. Program and test in FlashDevelop

4. Upload to Dropbox

Getting Started

*As of 9-7-13, Alex is still working on getting FlashDevelop to work with Apollolumi’s SWC. As a result, you won’t be able to work on anything yet unless you have the actual Flash CS6 program. In the meanwhile, you should still be able to look at the source code and do some AS3 practice if you’re unfamiliar with the language.*

You’ll want to download FlashDevelop. I suggest following this guide:

<http://creative-geeks.com/blog/2012/06/24/quick-tip-creating-and-compiling-an-as3-project-in-flashdevelop-without-the-flash-ide/>

If you’ve never programmed in AS3 before, fear not. It’s very similar to Java or C++. There are many tutorials online; here’s one I found in a few seconds that looks decent:

<http://prototyprally.com/making-games-in-actionscript-3-using-flashdevelop-part-1-the-setup/>

There are really just a few AS3-specific things you might want to know:

* MovieClips
  + Attributes such as x, y, rotation
* Coordinate systems, Point
  + Understanding localToGlobal(...) and globalToLocal(...)
* EventListeners

Important classes to look at:

* ContainerGame.as Main game controller
* Floater.as Main game abstract object
* Manager.as Keeps track of and controls Floaters

Game Structure

Extends Container.

Can be one of

ContainerPreloader.as

ContainerMenu.as

ContainerIntro.as

ContainerGame.as

ContainerOutro.as

One more Container, ContainerCutscene, will probably be included later on as well.

This MovieClip “holds” all of the currently displayed content.

The Engine controls when to destroy the current container and which container to create and display next. It also calls step(...) for everything.

Stage

Engine.as

Container.as

We’re most interested in ContainerGame.as.

ContainerGame.as

“cont”

“game”

magic

(Each black rectangle is a MovieClip.)

Basically, all interactive entities in the game are placed in a MovieClip called “cont”, which is inside another MovieClip called “game”, which is all inside ContainerGame.

Object Inheritance

Floater.as

abstract

Asteroid.as

DebrisFX.as

FXExplosion.as

Mineral.as

extends

PointDefenseLaser.as

ForceDefense.as

CommandCenter.as

EnemyKami.as

abstract

Enemy.as

abstract

Structure.as

Any object in the game that floats around extends Floater. This class contains the code needed for basic movement, collisions, etc. Simple objects extend only Floater, such as Asteroid. More complicated objects extend one more class, Structure or Enemy.

Additionally, each object is managed by a class extending Manager. All Managers are controlled by ContainerGame. These Managers are in charge of tasks such as stepping their objects and destroying them as appropriate.

For instance, AsteroidManager manages all Asteroids.

(P.S: There is no real way to declare a class as “abstract” in AS3.)

What to do?

First look at the implemented turrets and enemy (Point Defense Laser, Force Defense, and Kamikaze) and understand how they work. Also look at the classes they extend to see what methods you can use from them.

After claiming an idea from the list below, or coming up with an idea and getting it looked at by Alex, extend either Enemy.as or Structure.as and start programming!

*As of now, Alex is still trying to get FlashDevelop to work with Apollolumi’s SWC. So unfortunately, you won’t be able to test anything as of yet.*

< To be done after FlashDevelop decides to cooperate. >

Idea List

Basic Kamikaze

Smarter Kamikaze

Basic Shooter

Dodging and Shooter

Bomber

Stealth

Missile

Carrier

Dragger (tows away structures)

Interferer (blasts asteroids into small pieces)

Missile Turret

Railgun Turret

Blade Turret (Melee damage)

Attractor Turret

Heal Turret

Auto-Miner

Auto-Gatherer

Energy Turret (Boosts HP/Range/RoF/etc of nearby turrets)

Check the Dropbox list for which ones have an artist already working on it. It’s a lot better if there’s both 1 artist and 1 programmer working on the same enemy or turret.

Stretch

Implement armor and weapon types.

Asteroid types: Rock/Ferrous/Ice

Weapon types: Ballistic/Beam/Explosive

Also add Tech Tree. Glorious upgrades galore!

Artists

Primary Goal: Create vector art assets.

Process: 1. Claim or come up with an idea (idea list below)

2. Mail Alex for confirmation

3. Create pretty pictures

4. Upload to Dropbox

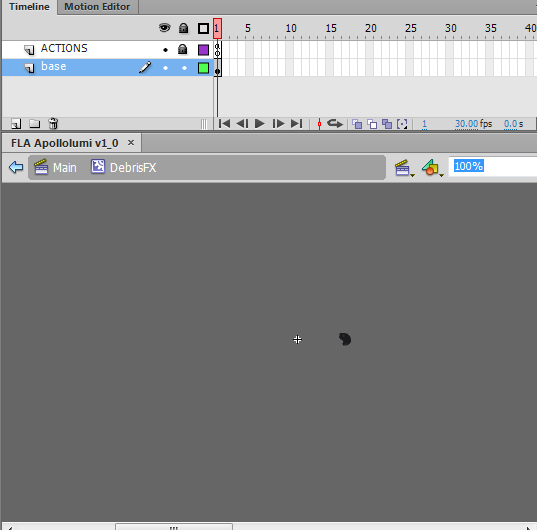
Getting Started

There are various categories to choose from, each with their own specifications.

*Debris*

Includes debris from destroyed enemies and structures.

These should be the easiest to make. All that’s needed is a vector image of the debris of your choosing.

Example:

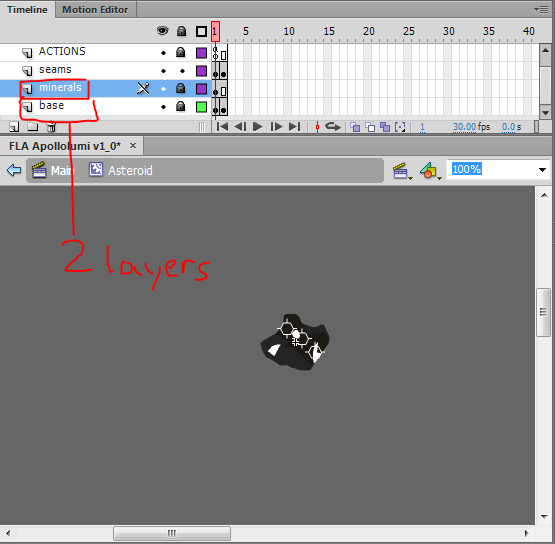
(actual size, shifted to the side so the registration cross isn’t in the way)

*Asteroids*

Includes big giant rocks that may or may not have juicy (?) minerals inside

A bit more involved. In addition to the actual asteroid, if you want the asteroid to have minerals, you’ll need to also provide a mineral layer. Make the minerals white (they’ll be tinted by Flash to the correct color).

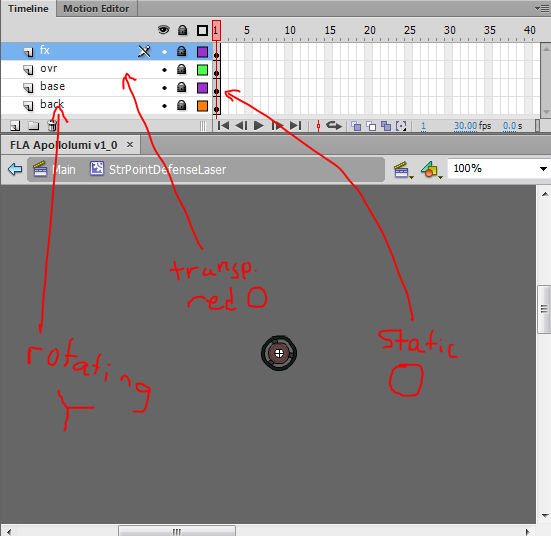
Also indicate what size you want the asteroid to be (1 is smallest; no max yet; each time an asteroid is broken, it spawns 2 asteroids of size n - 1 unless it was of size 1). Optionally, indicate another layer of “x”s or some other mark for where you want the seam targets to be.



*Enemies and Structures*

Check the Dropbox list for which ones have a programmer already working on it. It’s a lot better if there’s both 1 artist and 1 programmer working on the same enemy or turret.

More involved if you want stuff to move and junk. Include any moving parts (preferably only a rotation animation) on separate layers. Basic flashing lights, etc. are OK; put those on another layer, too. (I’ll put everything together in Flash, unless you have Flash, in which case just make your stuff inside a MovieClip.)



*Backgrounds*

Differs from all other assets in that they should be more realistic, not vectory.

Base backgrounds should be at least 800x600 and tileable. Special features, such as distant planets or nebulae, should have transparency, preferably as a .png. These don’t need to be tileable.

*Animations*

Make ‘em in Flash and be sure to put EVERYTHING into ONE MovieClip! I should be able to just import your MovieClip into the game, add a few lines of control code, and be good!

Sound Artists

Primary Goal: Create BGM and SFX

Process: 1. Ah, screw it. Just make something cool.

2. Upload to Dropbox

Getting Started

I suppose I’d prefer if people started with BGM, please. A guide to the style I’m shooting for is:

*Ratchet & Clank Soundtrack: Gemlik Base, Oltanis Orbit*

<http://www.youtube.com/watch?v=Go4_grJPFiQ>

Right, so if I forgot anything, apologies. I kind of rushed this. \*whoops\*

Just mail me if you have any questions!