Iain Lobb
C.E.O. Dull Dude Ltd!

Creating Successful Flash Games

Design or Development?

- Designers do graphics and developers do code?
- But in games
  - artists do graphics
  - developers do code
  - designers make it fun!
- So maybe we need a better definition.
  - Developers solve technical problems
  - Designers solve human problems.
- So we all do a bit of both.

Design or Development?

- With experience and better tools, things move from the development side to the design side. Which makes the whole process more fun!
  - var sound:Sound = new ShotgunSound();
  - var transform:SoundTransform = new SoundTransform();
  - transform.volume = 0.5;
  - var channel:SoundChannel = sound.play(0, 1, transform);
- Vs.
  - SoundManager.play(ShotgunSound, 0.5);

What are games?

- Games are about the relationships between objects in space.
- … but videogames layer-on images, characters, narratives, sound and music.
- Historically, most games are multiplayer (chess, football etc).
- Videogames allow the computer to take on the role of opponent.
- … but multiplayer is still more fun!

Goals

- Collect objects
- Kill enemies
- Reach locations
- Match objects
- Solve puzzles
- Make friends (real or virtual)
- Pwn noobs
- Most games use a combination.

Constraints

- Time limit
- Health
- Lives
- Ammo
- Money
- XP / Experience
- Inventory limit
- Doors & Keys
- Enemies, obstacles & puzzles

Modes of interaction

- Control a single avatar
- Command multiple agents
- Manipulate objects
- Match rhythms
- Create entities

Input

- Flash sucks at input!
- Mouse
  - Left button, context menu, wheel.
  - sprite.mouseX and sprite.mouseY
  - No wrapping or moving the cursor.
- Keyboard
  - KeyboardEvent.KEY_DOWN and KeyboardEvent.KEY_UP
  - Limited simultaneous presses.
  - CTRL, ALT and SHIFT are a no-go area!

Gamepad.as

- Here I come to save the day!
- http://github.com/iainlobb/Gamepad
- Based on analog joystick input.
  - var gamepad:Gamepad = new Gamepad(stage, true);
  - character.x += gamepad.x * 5;
  - character.y += gamepad.y * 5;
  - if (gamepad. fire1) { fire() };
- Use arrow keys, WASD or custom settings.
- Duplicate keys and fire rate coming soon.

Mousepad.as

- Coming soon…
- For games where you move or aim with the mouse.
- Similar abilities to Gamepad.as, but for mouse.

Top-down View

- True top-down
  - Easy to rotate sprites.
  - Good for driving games etc.
- Zelda-style
  - Art is drawn with false perspective.
  - Better for characters.
  - … but you have to draw every angle.
- Movement can be:
  - compass-direction (for characters)
  - or tank-style (for vehicles)

Side-on View

- Great for showing humanoid characters.
- Typically involves jumping around implausible vertical spaces called “platforms”.
- Variable jumps.
- Double-jumps.
- In-air movement control.
- “Brawler” view like Final Fight or Castle Crashers is somewhere between Top-down and Side-on.

Isometric View

- Graphics are drawn from a fixed 45 degree angle.
- Good for Sim games, puzzles and driving games.
- Great for showing buildings – see eBoy.
- Confusing for player movement – which way is left?

First person View

- Either using a 3D engine or pre-rendered from a fixed perspective.
- Great for shooting, exploration and puzzle games.
- There are other views
  - Follow-cam (basically the same as first person)
  - 3rd person 3D camera (can track players)
  - etc

3D

- All games are 3D.
  - A top down game where you can jump.
  - A parallax background behind a platform game.
  - So my new game engine is 100% 3D.
- …but most aren't.
  - Gravity means we live on a 2D plane.
  - We need a definite idea of up and down.
  - 2.5D works really well.

3D

- 3D is really hard, but the basics are easy:
  - var scaleRatio:Number = focalLength / (focalLength + z3d);
  - y = baseY + ((-z3d + y3d) * scaleRatio);
  - x = baseX + ((z3d + x3d) * scaleRatio);
  - scaleX = scaleY = scaleRatio;
- Luckily libraries are available to make your life easier.
  - http://blog.papervision3d.org/
  - http://away3d.com/
- 3D art is more labour intensive to create, but is much more flexible. Animation is harder still.
- If you're serious, consider switching to Untiy3D.

Collision Detection

- Built in
  - sprite.hitTestPoint()
  - sprite.hitTestObject()
  - bitmapData.hitTest()
- Pythagoras
  - dx = sprite1.x – sprite2.x;
  - dy = sprite1.y – sprite2.y;
  - distance = Math.sqrt((dx*dx)+(dy*dy));
- Bounding box
  - if (bottom1 < top2) return false;
  - if (top1 > bottom2) return false;
  - if (right1 < left2) return false;
  - if (left1 > right2) return false;
  - return(true);

Collision Detection

- More advanced:
  - Line-line
  - Line-grid
  - Line-circle
  - Point-grid
  - Circle-grid
  - Point-polygon
  - Polygon-polygon
  - And many more!

Physics

- Gravity
  - speedY += gravity; y += speedY;
- Bouncing
  - speedY *= -0.9;
- Beyond that it gets really, really hard!
- Luckily there are libraries to help you
  - http://box2dflash.sourceforge.net/
  - http://code.google.com/p/glaze/
  - http://lab.polygonal.de/motor_physics/
  - http://www.jiglibflash.com/blog/

Tile Grids

- Present since the earliest games.
- A neat simplification for games.
  - var gridX:int = sprite.x / tileSize;
  - var gridY:int = sprite.y / tileSize;
  - var grid:Array = [[“air”, ”air”, “trap”, “platform”], [“air”, “platform”, “air”, “air”]];
  - if (grid[gridX][gridY] == “trap”) {killPlayer()};
- Helps with collision detection, path-finding, level design.
- Powerful but restrictive.

Blitting

- Working directly with image data rather than using the standard display list.
  - bitmapData.copyPixels()
  - bitmapData.draw()
  - Matrix transformations
  - “buffers”
  - graphics.beginBitmapFill()
  - graphics.drawTriangles()
- Can be faster, but harder to do rotations etc
- Libraries
  - http://flixel.org/
  - http://flashpunk.net/

Art

- Flash is very flexible and can handle a range of styles.
  - Hand-drawn
  - Pixel-art
  - Cell-shaded cartoon
  - Painted
  - 3D rendered
  - Photographic
- It normally doesn't hurt to make your game look like a game though.

Animation

- The best animation is done by hand.
- Hand-draw frames and position manually.
- Tweens are normally better off in code.
  - object.x += (target.x – object.x) * 0.3;
- Tweening libraries save you many hours of work.
  - http://www.tweenmax.com/

Tuning and polish

- Weapon strength, time limits, enemy speed etc in easily modifiable variables or XML.
- Jump height needs to match level design.
- Particle effects, flashes, tints, screen shake
- What happens when you walk into a wall? Do you slide, walk on the spot, flatten against it?
- Good usability on U.I. – pictorial not text.
- Freedom of movement, expressiveness, “feel”, playability

You need a hook

- You need something which is unique to your game.
- Hook, Unique Selling Point (USP), gimmick, or twist.
- Character, weapon, unit type, mode of interaction, game mechanic.
- Don't just make a clone with different graphics.
- If you're going to copy something, work out what about it you're actually trying copy.

Sound & Music

- Can turn a good game into a great game.
- Very clunky built-in support.
- Fade-in/out, pause, mute, loop, etc:
  - http://evolve.reintroducing.com/2008/07/15/as3/as3-soundmanager/
- For sound effects try:
  - http://www.soundrangers.com/
  - http://www.drpetter.se/project_sfxr.html
- Audacity, Sony Acid Music, Garage Band etc
- Record your own!

Instructions

- Players don’t want to read instructions.
- Animation works better.
- Walk-throughs or tutorials are even better.
- Best of all is no instructions needed.
- Keep the controls on screen all the time.

Saving

- Really easy!
  - sharedObject = SharedObject.getLocal("game");
  - sharedObject.data.score=9232;
  - sharedObject.flush();
- Use "slots" to allow multiple saves.
- Progress can also be saved to a server.
- Or via level codes.

A.I.

- Be pragmatic and come up with simple rules that work.
- You're not making chess computers here - don't code the perfect enemy.
- Enemies are generally slow and weaker than your player – even their bullets move slower!
- A* path-finding etc is hard, but examples are out there.

Multiplayer

- Easy to do 2 players on one computer.
- Networked multiplayer adds a new level of fun but greater complexity.
- Most solutions require you to have your own server:
  - http://www.electro-server.com/
  - http://www.smartfoxserver.com/
- But not all!
  - http://nonoba.com/
  - http://playerio.com/

Multiplayer

- Turn-based and "deterministic" games require little or no server code.
- Real-time games require some Java, JavaScript or C# code on the server.
- Server-code may also be needed to stop cheating.
- Multiplayer games are less common but command higher prices as they encourage repeat plays.

Business Models

- Sponsorship.
  - Portals pay to add their branding to your game and add links back to their site.
  - Exclusive or non-exclusive licenses.
  - "Site locks"
  - Performance deals.
  - £100 – £10,000 per game.
  - http://www.flashgamelicense.com/

Business Models

- Advertising.
  - In-game or in surrounding HTML
  - Typically on preloader.
  - Consider launching your own portal.
  - £ low
  - http://www.mochimedia.com/
  - http://www.cpmstar.com/
  - https://www.google.com/**adsense**/

Business Models

- Sales.
  - Full game sales - typically download executable version, sell via paypal etc (now iPhone, Android)
  - Microtransations – users pay small amounts for access to in-game items, characters and levels.
  - £0.10-£10 per sale.
  - http://www.jambool.com/
  - https://www.mochimedia.com/developers/coins.html

Business Models

- Client briefs.
  - They pay you money and you make what they tell you to.
  - Steady income but less creative.
  - No chance of getting rich.
  - Can still be cool: e.g. Zwok!, Meta4orce.
  - £1000-£100,000

Thanks!

http://blog.iainlobb.com

http://twitter.com/iainlobb

iainlobb@googlemail.com