Endless Runner

Sprite Animation

Introduction

- Problem driven
  - Still looking at problems and solutions
  - Problem solving techniques
- Animation in Unity
  - Creating animation clips
  - Animation curves
  - Animation states

Today's Problems

1. I want to have an **animated** player sprite in my endless runner game.
2. I want the animation to change i.e. running must be different to jumping.

Problem Solving Techniques (Problem 1).

- Divide and conquer
  - Break the problem into smaller problems
    a. How to animate an object with more than one sprite?
    b. How to animate (e.g. rotate) an object with only one sprite?
    c. How to change the speed of an animation?
- Simplest problem first
  - Lets start with running, this only requires changing the sprite.
  - Then rotate the bolder, this only requires movement.
  - We'll move on to jumping this requires changing the image and movement.

- Change the problem
    - I've already done this
    - I added in the rotating bolder to explore movement before attempting it along side changing the image.

Problem: How to animate an object with more than one sprite?

Motivation

- We have sliced a sprite sheet and now have a number of animations made up of parts of the original sprite sheet.
- We want to move though these at a rate which gives an effective animation
- Some animations will loop (running) others will not (jumping/sliding)

Solution(s)

- Create animations in Unity
  - Worksheet (PlayerSpriteAnimation)
- What do we mean by animation?

Aside: Animation

- "Animation is the process that produces the illusion of movement" [1]
    - Present two or more image fragments (frames or cells)
    - Displayed in rapid succession with subtle changes to their content our eyes register this as movement.
    - Not a mystical art, but a well established process

Aside: Animation (cont.)

- Characteristics
  - Motion lines
  - Motion angles
  - Key-frames and in-betweens
  - Weight and gravity
  - Flexibility
  - Secondary actions
  - Cycles and loops
  - Tempo

Motion Lines (*or natural path*)

- An invisible line created by an object as it performs a sequence of moments
- Small changes = smooth / large changes = dramatic
- Examples: Bullet - straight / bouncing ball - wavy
  - Must conform to objects' real world behaviour
- Feldman (2001) suggests following "centre of gravity"
  - Doesn't always work (e.g. humans - head).

Motion Angles

- Essentially the direction the sprite is pointing
- Changes with changes in motion
  - Can be used to enhance changes
    - E.g. a jet making a steep bank

Key Frames

- Extremes of motion
    - E.g. flapping of wings, kicking of legs etc.
- More key frames = smoother movement (less = more jerky)
- Arcade style games (2 - 6 key-frames).

In-betweens

- Animations between key frames
- Must be small motions to give smooth animation
    - Creating these is know as *tweening*
- Creating in-betweens is time consuming
    - Software allows you to copy last frame
        - Traditionally called *onion skinning*

Weight and Gravity

- Animation must take account of
    - Weight
        - Affects speed, heavy objects slow to accelerate/turn etc.
    - Gravity
        - More effect on heavy objects, coin will always fall faster than a feather.

Flexibility

- Lack of flexibility leads to 'stiff' looking animation
- Appropriate ranges of motion must be observed
- Some body parts lead the animation, others follow
    - e.g. throwing is lead by the arm

Secondary Actions

- These are the parts of an object which follow the primary motion
    - e.g. the hair/cape of a walking character

Cycles and Loops

- Cycle
    - Repetitive motions displayed by animated objects
- Loop
    - Repartition of a cycle to give continuous (or extended motion).

Tempo

- Real world objects move at different rates during their motion
    - e.g. Running / Jumping
- 12fps minimum for animation (15 minimum for arcade games)
- Few arcade games run at more than 30!

Aside: Animation (cont.)

- Major Arcade Game Animation Primitives
    - Cylindrical primitive
    - Rotational primitive
    - Disintegrational primitive
    - Colour flash primitive
    - Scissors primitive
    - Growing primitive
    - Shrinking primitive

Cylindrical Primitive

- Spinning of cylindrical objects
    - e.g. tyres, missiles, barrels etc..

Rotational Primitive

- Rotating of objects
    - Usually 0-360 in increments (over x sprites).
    - Unity allows us to rotate sprites, far to expensive for arcade games.

Disintegration Primitive

- Removing objects from the screen
- Common methods:
    - Melting (blend pixels from the top down)
    - Dissolving (remove a random pattern of pixels over time)
    - Colour fading (change colour so object blends into the background).

Quick Example

Donkey Kong

1942

Colour Flash Primitives

- Subtle/dramatic changes in colour
- Quick and dirty effects with only a few sprites
- Size, colours, etc.. key.

Scissors Primitive

- Used for everything from walking to creatures biting
- Animation starts closed and ends with fully open
    - Tweens can create smoother motion
    - Normally 2-4 frames
    - Usually do not loop

Growing / Shrinking Primitives

- Used for objects getting bigger/smaller (or closer/further away)
- Usually a constant scale (and multiples of two).

Aside: Animation (cont.)

- Minor Arcade Game Animation Primitives
  - Piston Primitive (usually used for machines)
    - Move up/down or left right when animated
  - Squeeze Primitive
    - Appears to close or compress something
  - Swing
    - Variant of the scissor, pendulum like swinging

- Minor Arcade Game Animation Primitives (cont.)
    - Slide
        - Excessive exaggeration of key frame
        - Used to create sliding/crawling motions
        - Looks effective despite not part of the body leaving the ground
    - Open/Closed
        - Two states, used for doors eyes etc.
    - Bounce
        - Travels back a forth between two end points
    - Stomp
        - Simple flailing motion (typically two sprites)
        - Think 'space invaders'

Aside: Animation (cont.)

- Complex Arcade Game Animation Primitives
  - The slinking primitive
  - The flying primitive
  - The walking primitive
  - The running primitive (humans)
  - The running primitive (animals)
  - The jumping primitive
  - The crawling primitive

Slinking

- Used for snakes worms
- Elements of Squeezing Primitive

Flying

- Complex, requires study of bird/insect to get right
- Cyclical
    - 12 frames for a realistic effect
    - 4 frames (taking the most exaggerated components) is effective.
- Tempo a key element
    - e.g. dragon fly (fast) vs albatross (slow)

Walking

- One of the most complex animations
  - Remains an 'acid test' for animators
- Involves legs, arms and head
- Simple
  - 2 frames (modified scissor)
- Complex
  - 12 frames

Running (humans/animals)

- More complex (especially for animals)
- Fortunately a well studied topic (books etc.)

Jumping

- Standing jump (simple)
  - Stand, bend, leap, land
- Simple - 2 frames (unrealistic/cartoons style)
- Complex - 10 frames (realistic human standing jump)

Crawling

- Used to fit into tight spaces and avoid obstacles
  - Bend, crouch, move (i.e. crawling motion)
- Complexity comes from *opposite action*
  - Legs push and arms pull
- Simple - 2 frames (Slide primitive)
- Complex - 11 frames

Quick Example

Cannon Fodder 2

Prince of Persia

How to animate (e.g. rotate) an object with only one sprite?

Motivation

- Some sprites are not animated (one image), but **move**, rotate etc. how can I animate these in Unity?

Solution(s)

- Create animations in Unity
  - Worksheet (PlayerSpriteAnimation)

How to change the speed of an animation?

Motivation

- Some animations are faster than others, depending on their source. As we're using animations from other sources we need to be able to adjust these.
- There was also some discussion of Tempo earlier, what if we ant to adjust the tempo of an animation.

Solution(s)

- Animation scaling
    - Worksheet (PlayerSpriteAnimation)
- Animation key-frame positioning
    - Worksheet (PlayerSpriteAnimation)

Problem Solving Techniques (Problem 2).

- Divide and conquer
  - Break the problem into smaller problems
    1. How do we get Unity to play and animation when running/jumping/sliding?
    2. How to we change between them?
- Simplest problem first
  - N/A
- Change the problem
  - N/A

How do we get Unity to play and animation when running/jumping/sliding?

Motivation

- Depending on whether the Ninja is running, jumping or sliding we want to change the animation.
- The animation for each will be distinct.

Solution

- Animations are just another asset,
- Unity allows us to add more than one animation made up of sprites
    - Worksheet (TheAnimator)

How to we change between them?

Motivation

- We have animations and transitions between them.
- We need to tell the Animator component when to change the animation (state)

Solution

- Add an Animator component - this presents the animations as an FSM.
    - Wait .. what's an FSM?

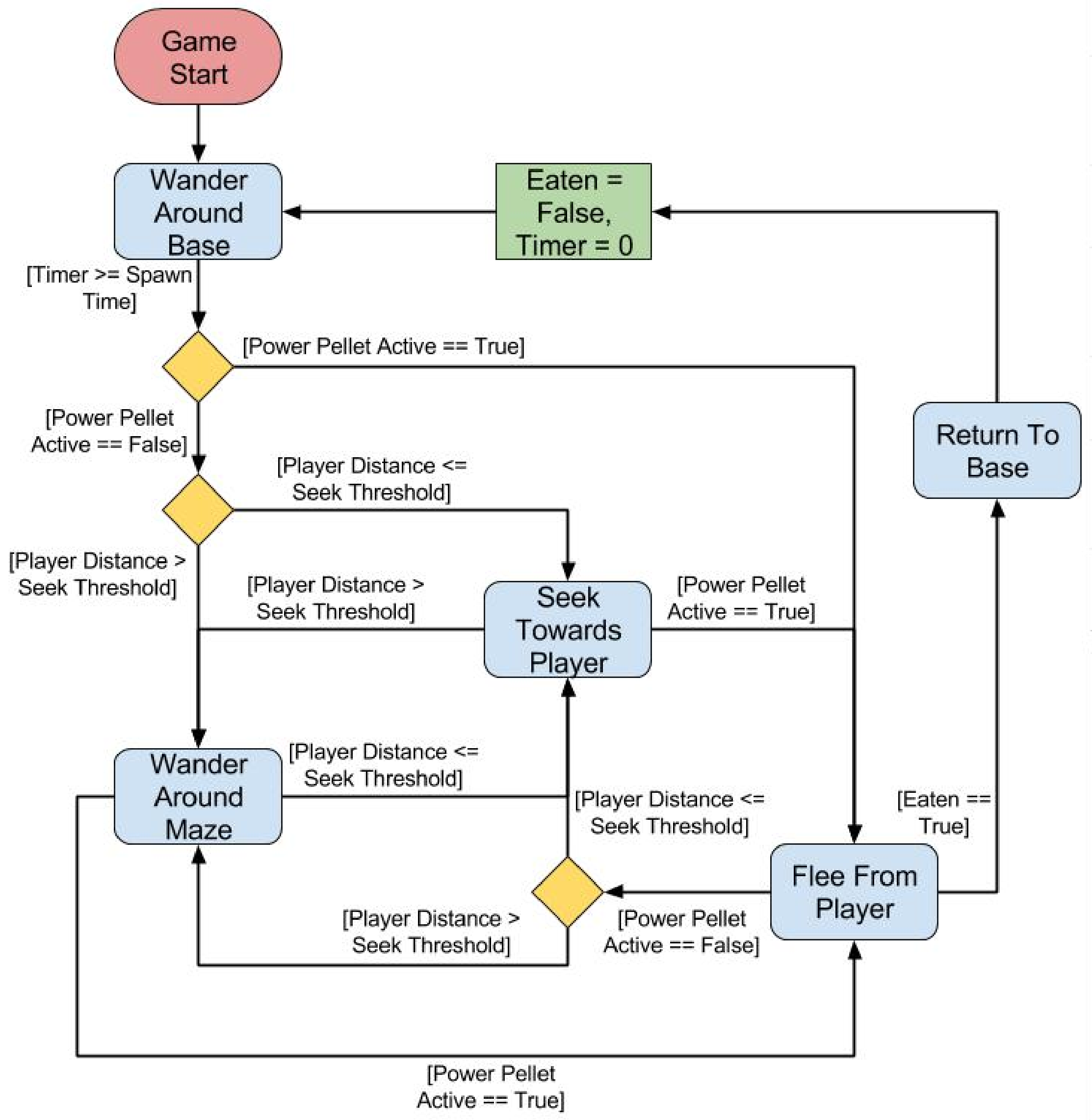
Aside: Finite State Machines (FSM)

- Definition: "A finite state machine is a device or a model of a device, which has a finite number of states it can be in at any given time and can operate on input to either make transitions from one state to another or to cause an output or action to take place. A finite state machine can only be in one state at any moment in time" [2]
- Big idea:
  - Make complex things simpler by dividing their behaviour into states
  - Ranges of parameters handled by a single state

Aside: FSM (cont.)

- Widely used:
    - Popular modelling tool in mathematics, computer science and other fields
- Games Examples:
    - Pack-Man
        - Evade & Chase states
        - Chase -> Evade (When player eats pill)
        - Evade -> Chase (When timer runs out)
    - Oversimplified?

This example takes the full life cycle of the ghost into account

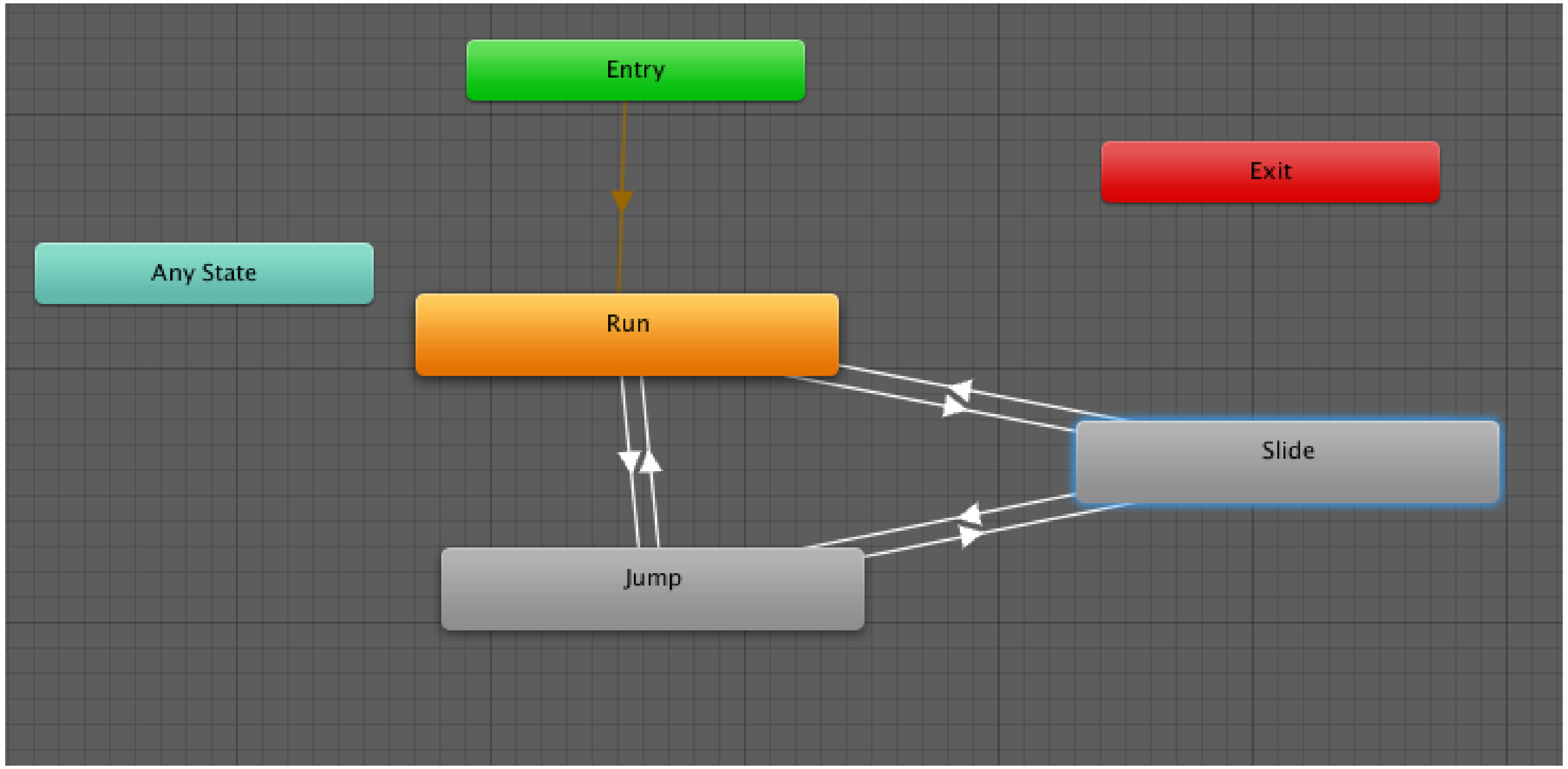

Game Start
Wander Around Base
Eaten = False, Timer = 0
[Timer >= Spawn Time]
[Power Pellet Active == True]
[Power Pellet Active == False]
[Player Distance <= Seek Threshold]
Return To Base
[Player Distance > Seek Threshold]
[Player Distance > Seek Threshold]
Seek Towards Player
[Power Pellet Active == True]
Wander Around Maze
[Player Distance <= Seek Threshold]
[Player Distance <= Seek Threshold]
[Eaten == True]
Flee From Player
[Player Distance > Seek Threshold]
[Power Pellet Active == False]
[Power Pellet Active == True]

Variable Name	Variable Definition
Spawn Time	The amount of time a ghost must wait before exiting the base
Timer	A timer on the ghost that is compared to Spawn Time
Power Pellet Active	Whether or not the player has just eaten a Power Pellet
Player Distance	The distance between the player and the current ghost
Seek Threshold	The maximum distance from the player the ghost can be to seek the player
Eaten	Whether or not a ghost has been eaten

Symbol	Definition
🟥	Terminal
🟦	State
🔶	Decision
🟩	Process
→	Transition
[asdfx]	Condition

Unity Animator:

This is the Unity Animator (with some states shown).



How do we get Unity to play and animation when running/jumping/sliding (cont.)

Recap: We have an animator, some states and we now want to move between them.

Solution (cont.)

- The Animator has Parameters, inputs it accepts whatever the state
- The transitions (arrows between states) have conditions
    - These describe the change in parameter state required to follow this state transition
- We then need some **code** to:
    - Take the input (W or S key)
    - Inform the Animator component attached to the player

Code

- Its C# (and its not scary)
- We add scripts like we've been adding components

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
```

Code (cont.)

```
public class PlayerMovement : MonoBehaviour
{
    private Animator animator;

        void Start ()
    {
        animator = GetComponent<Animator>();
    }

        // Update is called once per frame
        void Update () {

        }

    // Fixed Update called in sync with the physic engine (fixed rate per second).
    void FixedUpdate()
    {
        MoveCharacter();
    }
```

Code (cont.)

```
    void MoveCharacter()
    {
        if(Input.GetKey(KeyCode.W))
        {
            animator.SetBool("jump", true);
        }
        else
        {
            animator.SetBool("jump", false);
        }

        if (Input.GetKey(KeyCode.S))
        {
            animator.SetBool("slide", true);
        }
        else
        {
            animator.SetBool("slide", false);
        }
    }
}
```

Summary

Animation

- A number of primitive types are well established (for sprites)
- Some of these translate to 3D model animation

Animations in Unity

- Sprite animation using the sprite animator
- Can do limited movement/rotation over a frame
    - Might use this for rolling/shaking
    - Most movement is done but updating the position at intervals

PTO.

Next Week...

- Background
- Obstacles

References

[1] A. Feldman, Arcade Game Animation in Designing Arcade Computer Game Graphics, Worldware Publishing, Inc., Plano, Texas, 2001, pp. 293-356.
[2] M. Buckland, State-Drive Agent Design in Programming Game AI by Example, Worldware Publishing, Inc., Plano, Texas, 2005, pp. 43-84.