Case Study – 2D Platformer

Programming – Cross Platform Development

Last modified 10/02/16 by Sam Cartwright

Contents

- Types of 2D platformers
  - Action
  - Puzzle
  - Endless Runners
  - Metroid-Vania
- Player Controller
- Camera management
  - Anchors
  - Camera Box
  - Targets
  - Camera Smoothing

Types of 2D Platformer

- 2D platformers encompass a wide range of games
- Each kind of platformer needs to focus on different elements
  - Both in terms of design and implementation

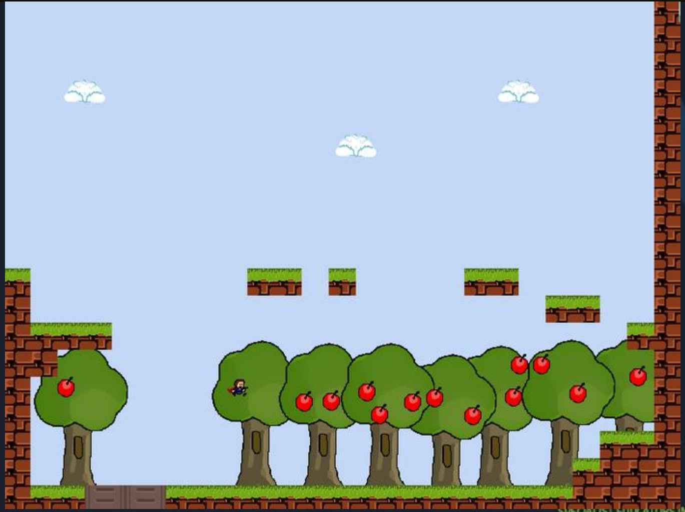
Types of 2D Platformer

- We will be talking about 3 major types of platformer
  - An overview of what that kind of game is about
  - Some of the programming requirements of each

- These distinctions are not absolute
  - Many games involve a mix of some or all of these

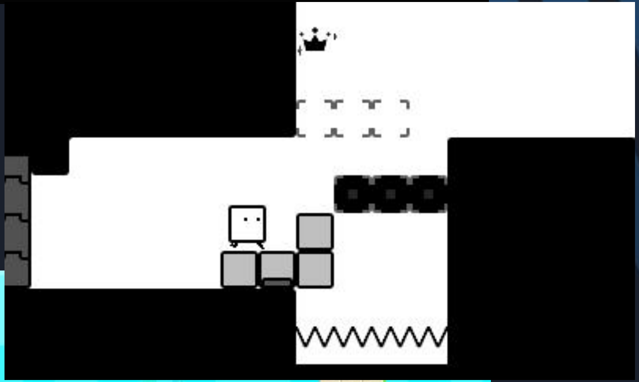
Action Platformers

- Often rely on the player making precise jumps, attacking enemies
- The game gives steadily harder hand-eye coordination challenges
- Controlling the player character is how the player shows mastery of the game
  - As such player controls are the most important aspect of the game
  - The player should always feel that a death was their fault and not the game



Puzzle Platformers

- Less about movement and control
- Often have a much broader set of mechanics
- By their nature, often have unique programming challenges



Endless Runners

- Player doesn't have direct control over player movement
- The level never ends and slowly gets harder until the player inevitably dies
- Requires procedural generation to continue forever
  - Typically implemented by randomly picking from a large number of pre-made chunks
  - Real random choice often doesn't feel good
  - Chunks start with a even chance to spawn
  - After spawning, their chance of appearing again goes down
  - Over time the chance of a chunk appearing goes up

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Metroid-Vania

- About exploring a large world
- The player finds upgrades that unlock now areas of the map
- Needs a level streaming system
  - Split the world into chunks
  - Keep the chunks closest to the player loaded in
  - Chunks tag what assets they need loaded in
  - As the player moves around, load in and out the needed chunks

Player Control

- The most important part of any platformer is the player control
  - Moving the player character is the main way the player interacts with the game world
  - Different kinds of platformers have different requirements for movement
  - In general, movement should be responsive
  - The player should never feel like they don’t understand how to move the player in a certain way

Player Physics

- You should not use a physics engine to drive your player
  - They don’t provide enough precise control to feel good
- Depending on the game there are several ways you could drive the movement of your player
  - Instant start and stop
  - Animation driven movement
  - Acceleration and drag

Collision Tips

- Typically we want to give the player leeway on collisions
  - Hitboxes for friendly things should be bigger
  - Hitboxes for enemies and harmful objects should be smaller

- Should have a small delay after leaving the ground before the player stops being able to jump
  - Very small – around 0.2 seconds
  - Smooths out noise in the ground collision
  - HDTVs have latency

Camera Management

- The camera in any game is very important

- The camera lets the player see the world

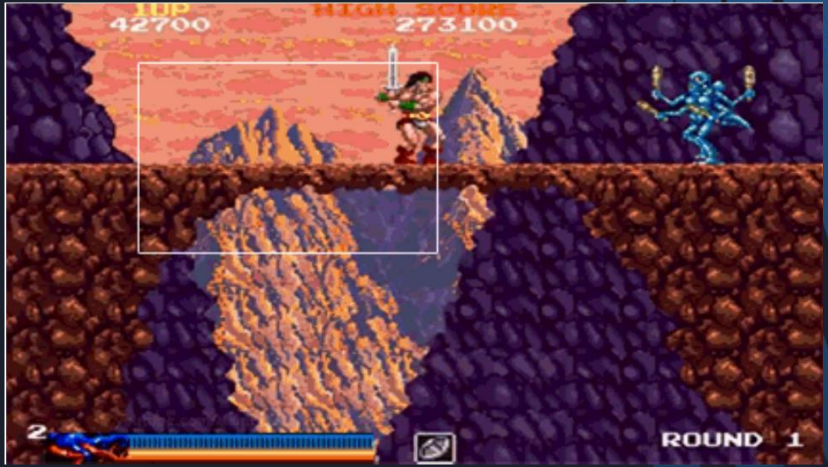
- There are many choices to make about how you might implement your camera

Position Locked

- Simplest kind of camera
- The Camera is locked directly to the player
- No other camera control is possible
- Simply set the camera position to the player every frame
- Having acceleration on the player can make this feel less jarring

Camera Box

- The player can move around freely in a small box



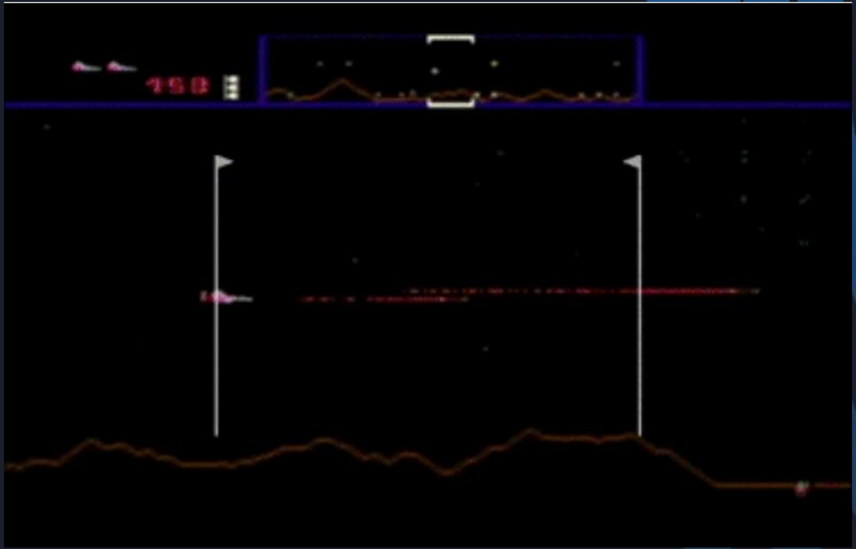
- Moving outside the box starts shifting the camera to keep the player inside the box

Camera Interpolation

- Setting the camera directly to a target can feel jarring
- Often its better to find a target point for the camera and smoothly move the camera towards it.
- A common way to do this is Interpolation
- pos = Lerp(pos, target, speed * dt)
- This leads to a nice smooth ease into the target position

Camera Anchors

- The camera tries to keep the player at a specific point on the screen
- What point the can be dependent on factors in the game
- Here we see two anchors that. The active one depends on what direction the player is facing



Camera Prediction

- The camera dynamically predicts where the player is going to go
  - Take the players position and add the players current velocity

- The camera tries to show where the player is going at all times

Summary

- 2D platformers encompass a wide range of game types
- Regardless of the type of game, player movement is one of the most important aspects
  - Don’t use a physics engine to drive the player
  - Give the player leeway, make sure the player feels like it was their own fault if they couldn’t make a jump
- There are many ways to drive cameras in side scrollers
  - Experiment and pick the one that best suits your game.

References

- Keren, Itay. 2016. *GDC Vault - Scroll Back: The Theory and Practice of Cameras in Side-Scrollers*. [ONLINE] Available at:http://www.gdcvault.com/play/1022244/Scroll-Back-The-Theory-and. [Accessed 10 February 2016].
- Monteiro, Rodrigo. 2016. *The guide to implementing 2D platformers | Higher-Order Fun*. [ONLINE] Available at: http://higherorderfun.com/blog/2012/05/20/the-guide-to-implementing-2d-platformers/. [Accessed 10 February 2016].