



## The Game Development Process: Level Design



## Selecting Features

- Note! First ...
  - Work on core mechanics (movement, shooting, etc.)
  - Get bugs worked out, animations and movement smooth
- Then, have
  - Prototype with solid core mechanics
  - Tweaked some gameplay so can try out levels
- Need
  - 25 levels!
  - Rest of features!
- Problem ... too many ideas!
  - If don't have enough, show it to some friends and they'll give you some

2



## Project 3 - Types of Features?

- Assume typical arcade-style game
- Player can use ...?
- Player must overcome ...?

3



## Project 3 - Types of Features

- Player can use
  - Abilities (attack moves, swimming, flying)
  - Equipment (weapons, armor, vehicles)
  - Characters (engineer, wizard, medic)
  - Buildings (garage, barracks, armory)
- Player must overcome
  - Opponents (with new abilities)
  - Obstacles (traps, puzzles, terrain)
  - Environments (battlefields, tracks, climate)
- Categorizing may help decide identity
  - Ex: Game may want many kinds of obstacles, or many characters.
  - What is *core*?

4



## Project 3 - Tips on Vetting

- Pie in the Sky
  - \*The Koala picks up the jetpack and everything turns 3d and you fly through this customizable maze at 1000 m.p.h.\*
  - Beware of features that are too much work
  - Don't always choose the easiest, but look (and think) before you leap
  - And don't always discard the craziest features ... you may find they work out after all
- Starting an Arms Race
  - \*Once the Koala's get their nuclear tank, nothing can hurt them. Sweet! No, wait ...\*
  - If you give player new ability (say tank) they'll like it fine at first
  - But subsequently, earlier challenges are too easy
  - You can't easily take it away next level
  - Need to worry about balance of subsequent levels
- One-Trick Ponies
  - \*On this one level, the Koala gets swallowed by a giant and has to go through the intestines fighting bile and stuff...\*
  - Beware of work on a feature, even if cool, that is only used once

5



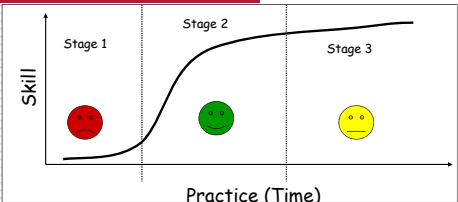
## Learning Curves?

- Practice versus Skill

6

**WPI**

## Learning Curves



- **Stage 1** – Players learn lots, but progress slow. Often can give up. Designer needs to ensure enough progress that continues
- **Stage 2** – Players know lots, increase in skill at rapid rate. Engrossed. Easy to keep player hooked.
- **Stage 3** – Mastered challenges. Skill levels off. Designer needs to ensure challenges continue.

7

**WPI**

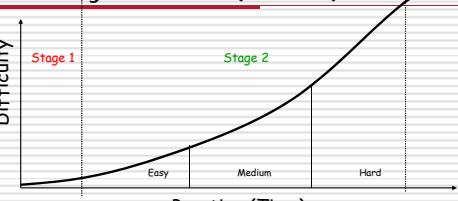
## Difficulty Curves?

- Practice versus Difficulty

8

**WPI**

## Difficulty Curves (1 of 2)

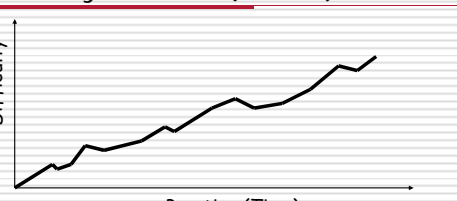


- Maintain **Stage 2** by introducing new features!
- Too steep? Player gives up out of frustration. Too shallow? Player gets bored and quits.
- How to tell? Lots of play testing! Still, some guidelines...

9

**WPI**

## Difficulty Curves (2 of 2)



- In practice, create a roller coaster, not a highway
- Many RPG's have monsters get tougher with level (*Diablo*)
  - But boring if that is all since will "feel" the same

10

**WPI**

## Project 3 - Guidelines

- Decide how many levels (virtual or real)
- Divide into equal groups of **EASY**, **MEDIUM**, **HARD** (in order)
- Design each level and decide which group
  - All players complete **EASY**
    - Design these for those who have never played before
  - Most can complete **MEDIUM**
    - Casual game-players here
  - Good players complete **HARD**
    - Think of these as for yourself and friends who play these games
- If not enough in each group, redesign to make harder or easier so about an equal number of each
- Have levels played, arranged in order, easiest to hardest
- Test on different players
- Adjust based on tests

11

**WPI**

## Make a Game that you Play *With*, Not *Against*

- Consider great story, graphics, immersion but only progress by trial and error ... is this fun?
- Ex: crossbowman guards exit
  1. Run up and attack. He's too fast. Back to save point (more on save points next).
  2. Drink potion. Sneak up. He shoots you. Back to save.
  3. Drop bottle as distraction. He comes looking. Shoots you. Back to save.
  4. Drink potion. Drop bottle. He walks by you. You escape!
  - Lazy design!
- Should succeed by *skill and judgment*, **not trial and error**
- Remember: Let the player win, not the designer!

Based on Chapter 5, *Game Architecture and Design*, by Rollings and Morris

12

## Specific Example - The Save Game Problem



- Should be used only so players can go back to their Real Lives™ in between games
  - Or maybe to allow player to fully see folly of actions, for exploratory and dabbling
- Don't design game around *need* to save
  - Has become norm for many games, but too bad
  - Ex: murderous level can only get by trying all combat options
- Beginner player should be able to reason and come up with answer
  - Challenges get tougher (more sophisticated reasoning) as player and game progress, so appeals to more advanced player
  - But not trial and error

Based on Chapter 5, *Game Architecture and Design*, by Rollings and Morris

13

## Different Level Flow Models



- Linear
- Bottlenecking
- Branching
- Open
- Hubs and Spokes

14

## Level Flow Model: Linear



- Start on one end, end on the other
- Challenge in making a truly interesting experience
  - Often try with graphics, abilities, etc.
  - Ex: *Half-life*, ads great story
- Used to a great extent by many games

15

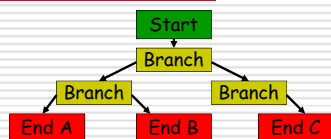
## Level Flow Model: Bottlenecking



- Various points, path splits, allowing choice
  - Gives feeling of control
  - Ex: Choose stairs or elevator
- At some point, paths converge
  - Designer can manage content explosion
  - Ex: must kill bad guys on roof

16

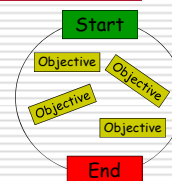
## Level Flow Model: Branching



- Choices lead to different endings
- User has a lot of control
- Design has burden of making many interesting paths
  - Lots of resources

17

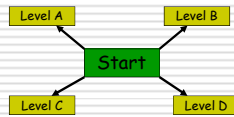
## Level Flow Model: Open



- Player does certain number of tasks
  - Outcome depends upon the tasks.
- Systemic level design
  - Designer creates system, player interacts as sees fit
- Sometimes called "sandbox" level. (Ex: GTA)

18

## Level Flow Model: Hub and Spokes



- Hub is level (or part of a level), other levels branch off
  - Means of grouping levels
- Gives player feeling of control, but can help control level explosion
- Can let player unlock a few spokes at a time
  - Player can see that they will progress that way, but cannot now

19

## Designing a Level: Brainstorming



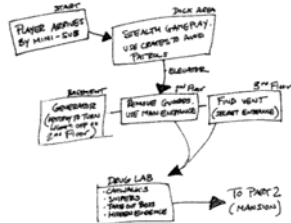
- An iterative process
  - You did it for the initial design, now do it for levels!
- Create wealth of ideas, on paper, post-it notes, whatever
  - Can be physical sketches
- Can include scripted, timed events (not just gameplay)
- Output
  - Cell-diagram (or tree)

20

## Designing a Level: Cell Diagram



- String out to create the player experience
- Ordered, with lesser physical interactions as connectors (i.e., hallways)



21

## QuakeII-DM1: An Example



- Video (Q2DM1\_Layout.avi)
  - level layout



22

## QuakeII-DM1: Architecture



- Two major rooms
- Connected by three major hallways
- With three major dead-ends
- No place to hide
- Forces player to keep moving
  - Camping is likely to be fatal

23

## QuakeII-DM1: Placement



- Cheap weapons are easy to find
- Good weapons are buried in dead ends
- Power-ups require either skill or exposure to acquire
- Sound cues provide clues to location
  - Jumping for power-ups
  - Noise of acquiring armor
- Video (Q2DM1\_Weapons.avi)
  - Weapon placement

24

## QuakeII-DM1: Result



- A level that can be played by 2-8 players
- Never gets old
- Open to a variety of strategies

25

## 5 Card Dash



- The designer's challenge
- Devise a sequence of levels that makes the player feel successful
- AND challenged
- WITHOUT losing them to boredom or frustration
  - Remember *Flow*?



A casual game  
• Poker crossed with Tetris  
• Video  
(5CD\_Intro.avi)

26

## 5 Card Dash Levels (1 of 2)



- Level 1: introduce the concept
  - Easy minimum hand
  - Easy required hands
  - Add some prompts along the way -- but not all at once
- Level 2:
  - More prompts with new features
  - Still easy

27

## 5 Card Dash Levels (2 of 2)



- Level 3
  - Add wildcards
  - Prompt bonus cards
  - Teach a straight
- Level 8
  - Prepare for level 9
- Level 9
  - Same as 8, but:
    - facedown cards
    - sequential goal
- Video (5CD\_Level9.avi)

28

## Heuristics for Level Design (1 of 2)



- Figure out what you're trying to "teach"
  - Make sure the level design expresses a need for that skill
- Provide incentives for the "right" behavior
  - Powerups, weapons, etc.
- Keep *Flow* in mind
  - Don't introduce too much at one time
  - Let people practice skills from time to time

29

## Heuristics for Level Design (2 of 2)



- Design for the game's features and capabilities
  - If you introduce, say, a new sniping weapon
    - Give it a long-distance target to practice on immediately
    - Create a level where it's the most important weapon
    - Then it's available to the player as a standard tool
  - If the engine bogs down in large outdoor areas...don't design one!

30

## Group Exercise

---

- Consider this classroom as a physical level
- Items:
  - Pages – players try to collect
  - Police – make player sit down for some time if caught
  - Detention chair – place where must sit if caught
  - Desks - obstacles
  - Power ups - various
- Design...