Interactive Media and Game Development

Game Design

Outline

- Selecting Features (next)
- Level Design
- Core Design

What Next?

- 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

Selecting Features – Types

- 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*?

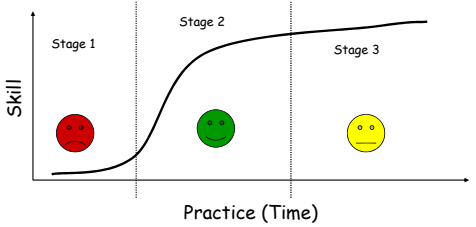
Tips on Vetting

- 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

Outline

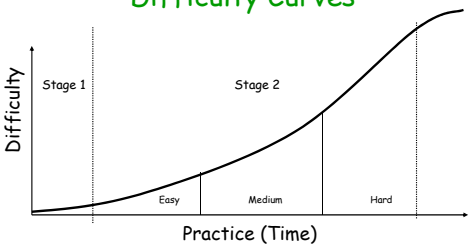
- Selecting Features (done)
- Level Design (next)
- Core Design

Learning Curves



- *Stage 1* – Players learn lots, bug 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.

Difficulty Curves



- 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…

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 of this genre
  - Good players complete HARD. These are designed for yourself and friends who play these games.
- If not enough in each group, redesign to make harder or easier so about equal number
- Play all and arrange in order, easiest to hardest
- Test on different players (friends and family, but enough in each category)
- Tweak according to outcomes of test

Outline

- Selecting Features (done)
- Level Design (done)
- Core Design (next)

Implementing Gameplay (1 of 2)

- Choices must be non-trivial, with *upside* and *downside*
  - If only upside, AI should take care of it
  - If only downside, no-one will ever use it
- Note, this is only regarding game theory
  - Ex: Could have ray gun that plays music. "Cool", but soon "gimme the BFG"
  - Ex: Nintendo's Smash Bro's has "Taunt" … ask: what for?
  - Ask: other examples from popular games?
- Gameplay value when upside and downside *and* payoff depends upon other factors
  - Ex: Rohan horsemen, but what if other player recruits pikemen?
  - Ex: Bazooka, but what if other player gets out of tank?

Implementing Gameplay (2 of 3)

- Should be *series* of interesting choices
  - Ex: Use of health potion now may depend upon whether have net for capturing more fairies
  - Having net may depend upon whether needed space for more arrows for bow
  - Needing arrows may depend upon whether killed all flying zombie bats yet
- Hence, well designed game should require *strategy*
- Game must display *complexity*
  - But doesn't mean it must be complex!
    - Don't make too many rules. Less if more.
    - Real world example: termites place one piece of mud. Results in hive, with cooling vents, etc.

Avoid Trivial Choices

- Horsemen → Archers → Pikemen
  - *Transitive,* not so interesting
- Horsemen → Archers → Pikemen → Horsemen (picture)
  - Ask: what game does this look like? (rock-paper-scissors)
  - *Intransitive,* more interesting
  - Ex: from LOTR Battle for Middle Earth
    - Horsemen fast, get to archers quickly with lances
    - Pikemen spears hurt horsemen bad
    - Pikemen slow, so archers wail on them from afar
- Don't want to hardwire. Sometimes A way better than B, sometimes a bit better, sometimes worse
  - The answer should depend upon the game situation, weather, terrain, time … also what opponent is doing

Ensuring Interesting Choices

- Interesting choices require good judgment on the part of the player
  - Correct choice must vary with circumstances
- Aim as designer, ensure circumstances don't stagnate and have only one right way to win
- No method for finding "best" choices
  - That's where creativity comes in (art)
- Still, some tips …

Toolbox of Interesting Choices

- Strategic versus Tactical
- Supporting Investments
- Versatility
- Compensating Factors
- Impermanence
- Shadow Costs
- Synergies

Strategic versus Tactical (1 of 3)

- Strategic choices affect course of game over medium or long term
  - *Tactical* choices apply right *now*
  - Ex: build archers or swordsmen (strategic)
  - Ex: send archers or swordsmen to defend against invading force (tactical)
- Strategic choices have effect on tactical choices later
  - Ex: if don't build archers, can't use tactically later

Strategic versus Tactical (2 of 3)

- Ex: *StarCraft*
  - Strategic choice: 1) upgrade range of marines, 2) upgrade damage, or 3) research faster fire
  - Which to choose?
    - If armored foes, Protoss Zealot, more damage
    - If fast foes, Zerglings, maybe faster fire
  - Other factors: number of marines, terrain, on offense or defense

Strategic versus Tactical (3 of 3)

- Ex: *Warzone 2100* (ask: who played?)
  - Build factories to spawn war machines
  - If build in level, then spawn quickly but factory only used for that level
  - If build at base, spawn slowly (have to ship to front lines) but factory can be used in subsequent levels
- Lesson: Good gameplay should have different choices leading to different *kinds* of payoff
  - Reduces the risk of trivial choices
  - Increase scope for good judgment

Supporting Investments

- Often game has primary goal (ex: beat enemy) but secondary goals (ex: build farms for resources)
- Some expenditures directly impact primary goal (ex: hire soldier), while others indirect (ex: build farm) called *supporting investments*
- Primary goals are "one-removed"
  - Ex: improve weapons, build extra barracks
- Supporting goals are "two-removed"
  - Ex: build smithy can then improve weapons
  - Ex: research construction lets you build smithy and build barracks (two and three removed)
    - Most interesting since strategic
- Payoff will depend upon what opponents do

Versatility (1 of 2)

- Rule of thumb is to ask what is best and worst about choices:
  1) This move does most damage, but slowest
  2) This move is fastest, but makes defenseless
  3) This move best defense, but little damage
  4) This neither best nor worst, but most versatile
- Most should be best in some way
- Versatile good for
  - beginners
  - flexibility (against unpredictable or expert opponent)

Versatility (2 of 2)

- Ex: beam can mine asteroids and shoot enemies
  - Versatility makes it good choice
- Speed is common way for versatility
  - Don't make fast units best
- If a versatile unit is also cheapest and most powerful → no interesting choice
  - (See "Compensating Factors", next)

Compensating Factors

- Consider strategy game where all units impeded by some terrain
  - Ships can't go on land, tanks can't cross water, camel riders only in dessert
- Assume flying unit that can go anywhere (Ask: how to balance?)
  1) Make slow
  2) Make weak, easily destroyed
  3) Make low surveillance range (unrealistic)
  4) Make expensive
- Note, last choice common but uninteresting since doesn't change tactical use
- Choice should be clear to player. Don't make a gamble before they know.
  - Ex: pick troops (cold weather) then find in jungle

Impermanence (1 of 2)

- Some permanent (ex: you get to treasure first), others not (ex: I got storage near mine, but you can grab it off me)
- Really, another kind of compensating factor
  - I.e. – impermanence can compensate for something being really good
- Can be used for interesting choices
  - Ex: choice of medium armor for rest of game or invulnerable for 30 seconds?
- Advantage (or disadvantages) can be impermanent in number of ways:

Impermanence (2 of 2)

- (Examples mostly from *Magic the Gathering – Battlegrounds*)
  - Can be destroyed (enchantments, ex: *gratuitous violence* makes units tough, but can be destroyed)
  - Can be stolen or converted (ex: *threaten* steals or converts enemy for short time)
  - Can be applied to something you don't always have (ex: *goblin king* gives bonus to goblins, but must have goblins)
  - Certain number of uses (ex: three grenades, but grenade spamming)
  - Last for some time (wears off, ex: Mario *invulnerable star*)
- Common in games, but deserves special attention

Shadow Costs (1 of 2)

- In a game, continually presented with costs and trade-offs. But not all direct.
  - Ex: soldiers for gold, but need armory first for weapons and barracks for soldiers
  - Called *shadow costs* for supporting investments
  - Can make flow chart mapping shadow costs

Shadow Costs (2 of 2)

- Ex: Age of Mythology has wood and food. Food is inexhaustible, wood is finite
  - Charioteer
    - Costs 60 wood, 40 food and 40 seconds to spawn
    - Shadow costs vary over game
      - Early on, food and wood expensive, spawn doesn't matter
      - Mid-game, much food and wood, spawn makes it harder to pump out new units
      - End-game, no wood, spawn is priceless
- Use variability to add subtlety to game. Vary environment and vary shadow costs (ex: more trees to vary cost of wood)
  - Challenge for level designer
  - Expert players will appreciate

Synergies (1 of 2)

Synergies are interaction between different elements of player's strategies (note, terms may be different than ch 2.2)

- Positive Feedback
  - Economies of Scale – the more of one type, the better (ex: wizards draw strength from each other)
  - Economies of Scope – the more of a set, the better, or advantage of combined arms (ex: trident and net, infantry and tanks)
- Negative Feedback
  - Diseconomies of scale – first is most useful, others have less benefit (ex: diminishing returns from more peasants entering a mine since get in each other's way)
  - Diseconomies of scope – (ex: mixed troops go only as fast as slowest)

Synergies (2 of 2)

- Ideally, all go together at once, but can emphasize
  - Ex: Chess is a game of positive feedback
  - Small advantage early on, exploited to crushing advantage
- Game of negative feedback needs other ways to keep interesting
  - Ex: trench combat makes a "catch-up" factor, or as get far from base, supply long grows, game lasts a long time
  - Ex: *Super NES NBA Jam* – catch up setting as an equalizer
- Be aware of each

Review: Use Tools from Toolbox of Interesting Choices

- Strategic versus Tactical
- Supporting Investments
- Versatility
- Compensating Factors
- Impermanence
- Shadow Costs
- Synergies

- Groupwork:
  - Use 1-2 in a game about graduating from high school. Discuss.

AI and Games

- Opponents that are challenging, or allies that are helpful
  - Unit that is credited with acting on own
- Human-level intelligence too hard
  - But under narrow circumstances can do pretty well
  - Ex: chess and Deep Blue
- Artificial Intelligence
  - Around in CS for some time
  - Games a special niche (needs to be real-time)

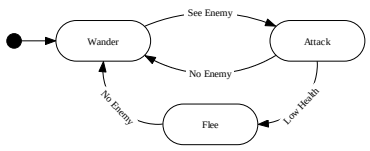
AI and Games

- Must be smart, but purposely flawed
  - Lose in a fun, challenging way
- No unintended weaknesses
  - No "golden path" to defeat
  - Must not look dumb
- Must perform in real time (CPU)
- Configurable by designers
  - Not hard coded by programmer
- "Amount" and type of AI for game can vary
  - RTS needs global strategy, FPS needs modeling of individual units at "footstep" level
  - RTS most demanding: 3 full-time AI programmers
  - Puzzle, street fighting: 1 part-time AI programmer

Group Exercise

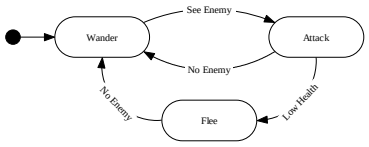
- Consider game where hero is in a pyramid full of mummies. Mummy – wanders around maze. When hero gets close, can "sense" and moves quicker. When it can see hero, rushes to attack. If wounded, flees.
- What "states" can you see? What are the transitions? Can you suggest Game Maker appropriate code?

Finite State Machines (1 of 2)



- Abstract model of computation
- Formally:
  - Set of states
  - A starting state
  - An input vocabulary
  - A transition function that maps inputs and the current state to a next state

Finite State Machines (2 of 2)



- Most common game AI software pattern
  - Natural correspondence between states and behaviors
  - Easy to understand, program and debug
  - Completely general to any problem
- Problems
  - Explosion of states
  - Often created with ad-hoc structure