IMGD 1001:
Game Balance

Outline

- □ Gameplay (done)
- □ Level Design (done)
- □ Game Balance (next)

Mini-Outline

- □ Broadly, game balance includes:
  - ■ Player-Player (next)
  - ■ Player-Gameplay
  - ■ Gameplay-Gameplay

Player/Player Balance (1 of 2)

- □ Players should have “fair” chance of winning
  - ■ Advantage only in skill
  - ■ Any luck should be infrequent, minor and equal to both
- □ Ex: *Virtua Fighter*
  - ■ Say, Sarah Bryant beats Lion every time.
  - ■ Does that mean unbalanced?
    - □ Not necessarily, look more closely
- □ Suppose friend said could beat everyone as Sarah Bryant all the time.
  - ■ Would only be a problem if beginner as Sarah always beat expert as Lion
  - ■ And if could choose characters? Sarah versus Sarah?

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

Player/Player Balance (2 of 2)

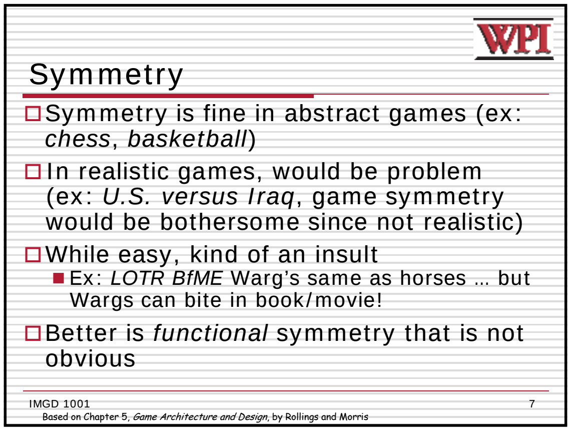
- □ Allow to arrange victory by *skill and judgment*
- □ Avoid results mostly as stroke of luck
  - ■ Right from the start or magnified as game progresses (ex: start close to gold mine provides escalating advantage)
- □ Simplest way is to have symmetry
  - ■ Same weapons, maneuvers, hit points (sports do this – teams are nearly always symmetric)
  - ■ But note, not always the most interesting. Want different moves on fighters, say. (More later)

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

Symmetry - Example

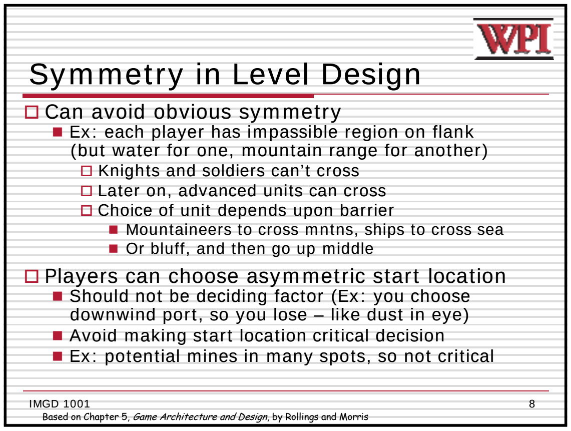
- □ Two heroes square off for duel, poised in kung fu stances. Both are equally matched.
- □ They wait for an advantage.
- □ Hours pass. Days pass.
- □ Breeze comes by, flicks spec of dust in one’s eye
- □ Blinks, frowns then bows
- □ Know result without fight → tiny asymmetry enough to decide outcome!
- □ If breeze or dust decided game, is that ok?
  - ■ No … you’d want your money back!
- □ Don’t want to decide by factors out of user control
  - ■ Keep symmetric

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



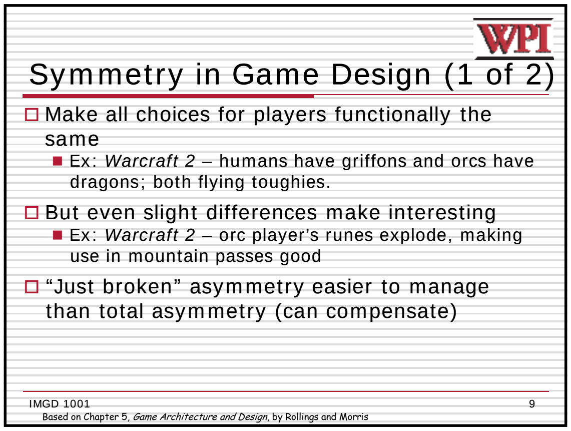
Symmetry

- □ Symmetry is fine in abstract games (ex: *chess*, *basketball*)
- □ In realistic games, would be problem (ex: *U.S. versus Iraq*, game symmetry would be bothersome since not realistic)
- □ While easy, kind of an insult
  - ■ Ex: *LOTR BfME* Warg's same as horses … but Wargs can bite in book/movie!
- □ Better is *functional* symmetry that is not obvious



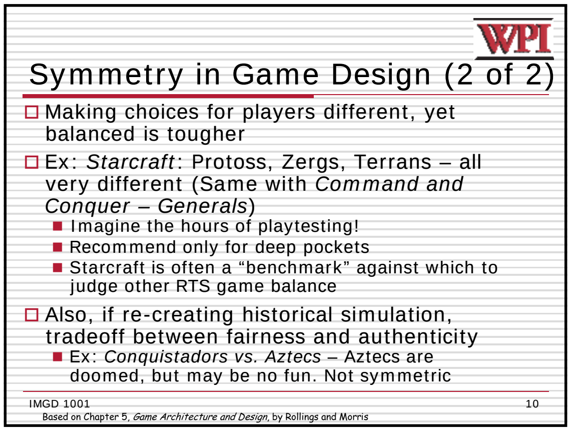
Symmetry in Level Design

- □ Can avoid obvious symmetry
  - ■ Ex: each player has impassible region on flank (but water for one, mountain range for another)
    - □ Knights and soldiers can't cross
    - □ Later on, advanced units can cross
    - □ Choice of unit depends upon barrier
      - ■ Mountaineers to cross mntns, ships to cross sea
      - ■ Or bluff, and then go up middle
- □ Players can choose asymmetric start location
  - ■ Should not be deciding factor (Ex: you choose downwind port, so you lose – like dust in eye)
  - ■ Avoid making start location critical decision
  - ■ Ex: potential mines in many spots, so not critical



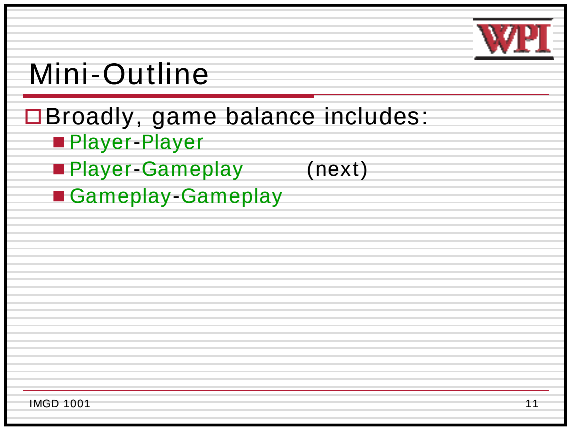
Symmetry in Game Design (1 of 2)

- □ Make all choices for players functionally the same
  - ■ Ex: *Warcraft 2* – humans have griffons and orcs have dragons; both flying toughies.
- □ But even slight differences make interesting
  - ■ Ex: *Warcraft 2* – orc player's runes explode, making use in mountain passes good
- □ "Just broken" asymmetry easier to manage than total asymmetry (can compensate)



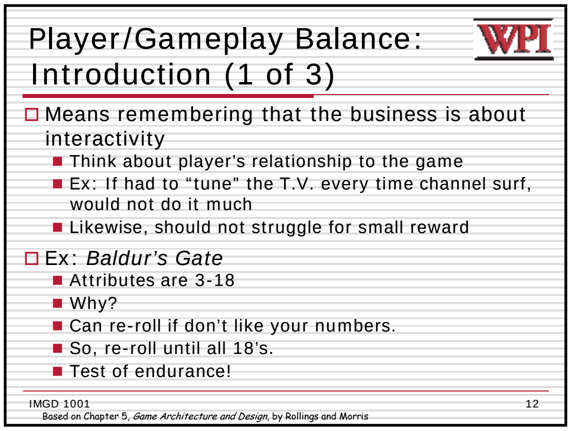
Symmetry in Game Design (2 of 2)

- □ Making choices for players different, yet balanced is tougher
- □ Ex: *Starcraft*: Protoss, Zergs, Terrans – all very different (Same with *Command and Conquer – Generals*)
  - ■ Imagine the hours of playtesting!
  - ■ Recommend only for deep pockets
  - ■ Starcraft is often a "benchmark" against which to judge other RTS game balance
- □ Also, if re-creating historical simulation, tradeoff between fairness and authenticity
  - ■ Ex: *Conquistadors vs. Aztecs* – Aztecs are doomed, but may be no fun. Not symmetric



Mini-Outline

- □ Broadly, game balance includes:
  - ■ Player-Player
  - ■ Player-Gameplay (next)
  - ■ Gameplay-Gameplay



Player/Gameplay Balance: Introduction (1 of 3)

- □ Means remembering that the business is about interactivity
  - ■ Think about player's relationship to the game
  - ■ Ex: If had to "tune" the T.V. every time channel surf, would not do it much
  - ■ Likewise, should not struggle for small reward
- □ Ex: *Baldur's Gate*
  - ■ Attributes are 3-18
  - ■ Why?
  - ■ Can re-roll if don't like your numbers.
  - ■ So, re-roll until all 18's.
  - ■ Test of endurance!

Player/Gameplay Balance: Introduction (2 of 3)

- □ Player/Gameplay balance entails balancing challenges against player's improvement curve
  - (We talked about this previously, see Gameplay slides with graphs)

Player/Gameplay Balance: Introduction (3 of 3)

- □ Often, have difficulty settings (player manually selects)
  - Still challenge of making the "Normal" level right.
- □ Compromises
  - Could ask player up front some questions (ex: have you played FPS before?), then recommend setting
  - Could have player do tutorial level, then recommend setting

Sub-Outline

- □ Again, true balance is an art, but three guidelines that can help
  1) Reward the player
  2) Let the machine do the work
  3) Make a game that you play *with*, not *against*

Reward the Player

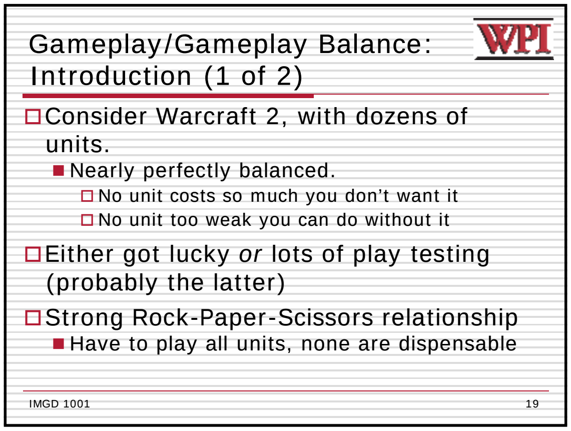
- □ Player will have to learn. Will make mistakes (discouraging)
- □ Want to offset with reward when they do something right
- □ Ex: *Virtua Fighter*, takes longer to learn complicated moves
  - Sarah's backflip. Reward comes from seeing flip (eye candy) and punch in kidneys (payoff)
- □ Best when expand game options
  - Ex: "Now with backflip, I can see new use for reverse punch"
- □ In general, better to reward player for something *right* than punish for something *wrong*
  - Punishment makes players not want to play

Let the Machine do the Work

- □ Interface should show player the world and let him/her manipulate
- □ Computer is tool to take care of wide-range of tedious tasks
  - If tasks are not fun, don't make player do them
- □ Blur of boundary between chore and game feature
  - RPG could provide graph so player can manually draw map as they explore … but is that fun?
  - Ex: In *D&D*, can tell D.M. "We go back to the dungeon entrance". Easy, fun. What if a game makes player walk back over map that has been seen? Boring, no fun.
  - Ex: *Myst* provides lightning bolt move to avoid tedium
  - Other examples?
- □ Also, if option is no-brainer, then AI should take care of it!

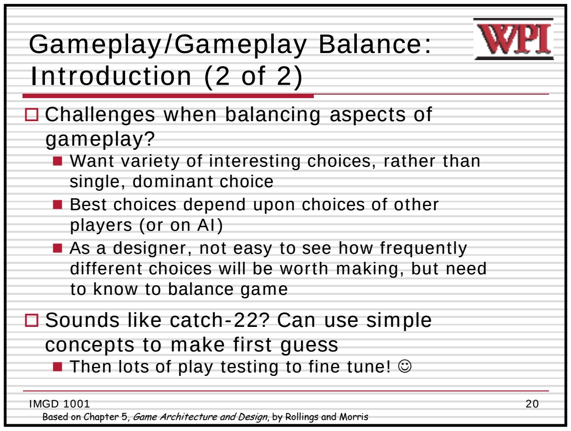
Mini-Outline

- □ Broadly, game balance includes:
  - Player-Player
  - Player-Gameplay
  - Gameplay-Gameplay (next)



Gameplay/Gameplay Balance: Introduction (1 of 2)

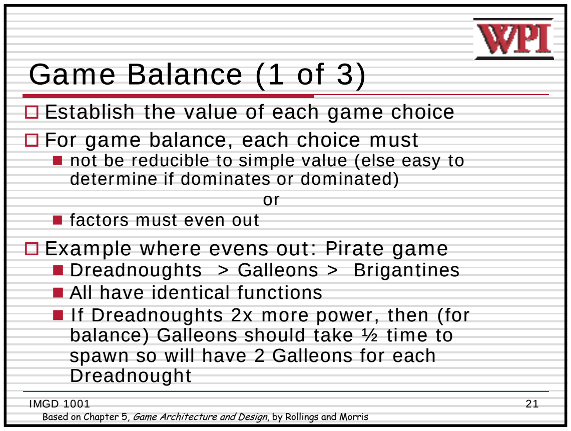
- Consider Warcraft 2, with dozens of units.
  - Nearly perfectly balanced.
    - No unit costs so much you don't want it
    - No unit too weak you can do without it
- Either got lucky *or* lots of play testing (probably the latter)
- Strong Rock-Paper-Scissors relationship
  - Have to play all units, none are dispensable



Gameplay/Gameplay Balance: Introduction (2 of 2)

- Challenges when balancing aspects of gameplay?
  - Want variety of interesting choices, rather than single, dominant choice
  - Best choices depend upon choices of other players (or on AI)
  - As a designer, not easy to see how frequently different choices will be worth making, but need to know to balance game
- Sounds like catch-22? Can use simple concepts to make first guess
  - Then lots of play testing to fine tune! ☺

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

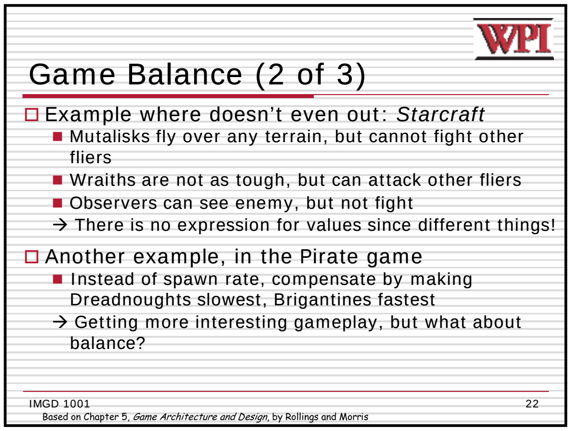


Game Balance (1 of 3)

- Establish the value of each game choice
- For game balance, each choice must
  - not be reducible to simple value (else easy to determine if dominates or dominated)

    or
  - factors must even out
- Example where evens out: Pirate game
  - Dreadnoughts > Galleons > Brigantines
  - All have identical functions
  - If Dreadnoughts 2x more power, then (for balance) Galleons should take ½ time to spawn so will have 2 Galleons for each Dreadnought

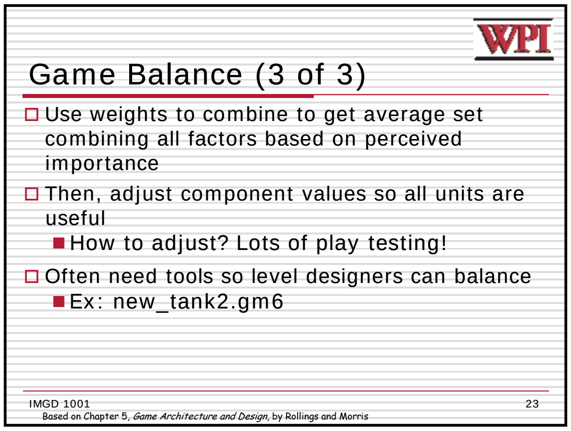
Based on Chapter 5, *Game Architecture and Design*, by Rollings and Morris



Game Balance (2 of 3)

- Example where doesn't even out: *Starcraft*
  - Mutalisks fly over any terrain, but cannot fight other fliers
  - Wraiths are not as tough, but can attack other fliers
  - Observers can see enemy, but not fight
  - → There is no expression for values since different things!
- Another example, in the Pirate game
  - Instead of spawn rate, compensate by making Dreadnoughts slowest, Brigantines fastest
  - → Getting more interesting gameplay, but what about balance?

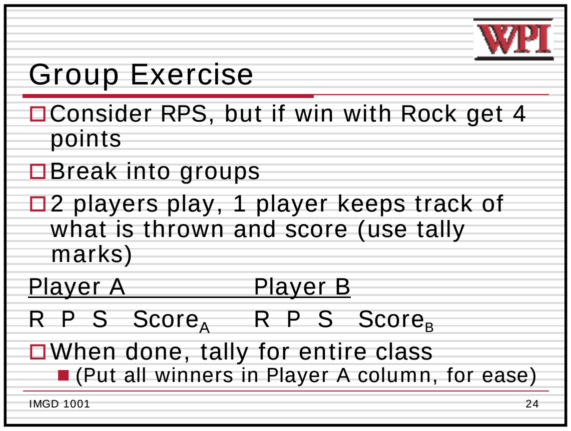
Based on Chapter 5, *Game Architecture and Design*, by Rollings and Morris



Game Balance (3 of 3)

- Use weights to combine to get average set combining all factors based on perceived importance
- Then, adjust component values so all units are useful
  - How to adjust? Lots of play testing!
- Often need tools so level designers can balance
  - Ex: new_tank2.gm6

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



Group Exercise

- Consider RPS, but if win with Rock get 4 points
- Break into groups
- 2 players play, 1 player keeps track of what is thrown and score (use tally marks)

Player A   Player B

R P S Score_A R P S Score_B

- When done, tally for entire class
  - (Put all winners in Player A column, for ease)

Intransitive Game Mechanics (1 of 3)

	Rock	Paper	Scissors
Rock	0	-1	+2
Paper	+1	0	-1
Scissors	-2	+1	0

(Ex: I choose scissors, you choose rock. You get +2, I get zero so difference is -2)

- □ Payoff Matrix
- □ Say payoff is R, P, S and frequency r, p, s
  - ■ Want to know how often used (r, p, s)

Intransitive Game Mechanics (2 of 3)

- □ Net payoff R is (0 x r) + (-1 x p) + (2 x s)
  1) R = -1p + 2s
  2) P = r – s
  3) S = -2r + 2p
- □ Sum must be zero (zero sum game, whatever one player gains other loses. Both cannot have net gain.)
  - ■ R + P + S = 0
- □ All net costs must be equal else would favor (remember, triangle example)
  - ■ R = P = S

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

Intransitive Game Mechanics (3 of 3)

- □ Solve:
  - ■ (eq2) r - s = -p + 2s (eq1)
    → r = -1p + 3s
  - ■ (eq2) -2(-1p + 3s) + p = -1p + 2s (eq1)
    3p – 6s = -p + 2s
    4p = 8s → p = 2s → r = s
  - ■ Since r + p + s = 1 (sum of probabilities)
    s + 2s + s = 1
    4s = 1 → s = 0.25, r = 0.25, s = 0.5
- □ Ratio → Rock and Scissors 25%, Paper 50%
  - ■ Probably not what expected.
  - ■ Often result … if one option more expensive, others are most affected

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

Combinatorial Explosions

- □ How many components should there be to make interesting?
  - ■ Too few? Then becomes trivial (Ex: in Hastings, only way to change power base is to put infantry on hill)
  - ■ Too many? Then too hard to have skilled play
- □ Rule of thumb: N factors that could modify core mechanics, and each boolean (hill or not, rain or not …) → 2^N possible combinations … explodes rapidly
  - ■ Remember, N=24 gives about 16 million combinations!
  - ■ Err on the side of caution

"In Populous (EA god-game), should have lots of characters or half-dozen? Noticed would be easier to understand game experience with few, versatile units rather than many specific ones."
Richard Leinfellner (executive in charge of *Bullfrog*)

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

Design Scalability

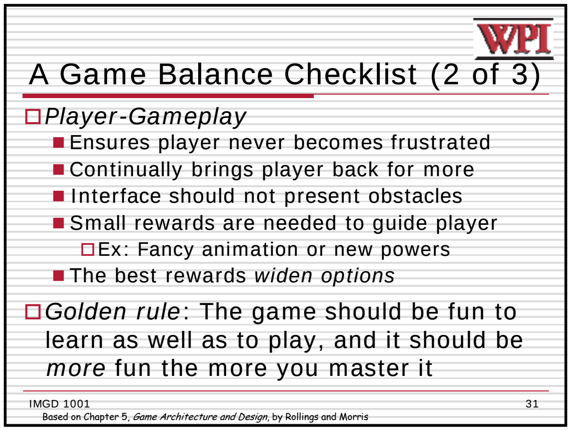
- □ Intransitive designs are inflexible
  - ■ If have balanced relationship and remove one, will have dominated strategy
  - ■ Ex: RPS and remove R … always choose S!
- □ If project lead says behind schedule, so don't include 5th orc type
  → Elegant design falls like a house of cards!
- □ But is relatively easy to add components
  - ■ Doesn't have to be symmetrical, can be redundant or useful in only a few cases
    - □ Ex: scout, or special spell
- □ Lesson
  - ■ If you are going to scale, scale ***up*** not ***down***

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

A Game Balance Checklist (1 of 3)

- □ *Player-Player*
  - ■ Ensures game is fair
  - ■ Especially important for multiplayer games
  - ■ Symmetry works for this, but asymmetry may be needed or more appealing (try "just broken")
  - ■ Make sure any asymmetry doesn't magnify imbalance as game progresses
- □ *Golden rule*: a player should never be put in an unwinnable situation through no fault of their own

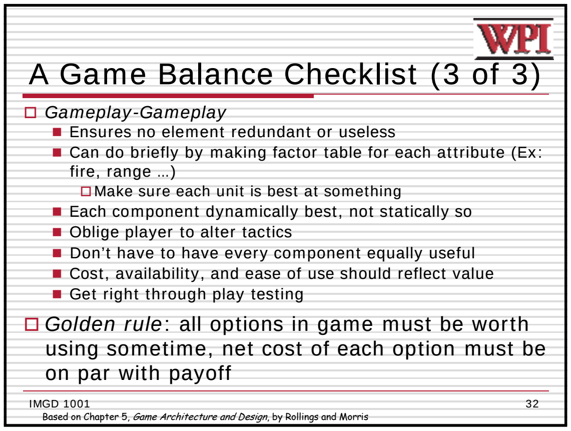
Based on Chapter 5, *Game Architecture and Design*, by Rollings and Morris



A Game Balance Checklist (2 of 3)

- *Player-Gameplay*
  - Ensures player never becomes frustrated
  - Continually brings player back for more
  - Interface should not present obstacles
  - Small rewards are needed to guide player
    - Ex: Fancy animation or new powers
  - The best rewards *widen options*
- *Golden rule*: The game should be fun to learn as well as to play, and it should be *more* fun the more you master it

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



A Game Balance Checklist (3 of 3)

- *Gameplay-Gameplay*
  - Ensures no element redundant or useless
  - Can do briefly by making factor table for each attribute (Ex: fire, range …)
    - Make sure each unit is best at something
  - Each component dynamically best, not statically so
  - Oblige player to alter tactics
  - Don't have to have every component equally useful
  - Cost, availability, and ease of use should reflect value
  - Get right through play testing
- *Golden rule*: all options in game must be worth using sometime, net cost of each option must be on par with payoff

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