Lecture 3

Design Elements
Reminder: Aspects of a Game

- **Players**: How do humans affect the game?
- **Goals**: What is the player trying to do?
- **Rules**: How can the player achieve the goal?
- **Challenges**: What obstacles block the goal?
Formal Design Elements

• **Players**: Player Mode Sketches

• **Goals**: Objectives

• **Rules**: Actions and Interactions

• **Challenges**: Obstacles and Opponents
Player Mode Sketches

- Game may have several *player modes*
  - Ways in which player interacts with a game
  - **Example:** Inventory screen vs. combat screen

- You should *storyboard* all of your modes
  - Sketches of each of the major player modes
  - May have action (like movie storyboard)
  - Illustrate how player interacts with game
Dragon Age: Standard Mode
Dragon Age: Inventory Mode

Design Elements
Aside: Help the Hero

When the hero finds an item, drag them with your mouse onto his inventory grid.

Use the arrow keys to rotate items. Combine certain items to save space!
Lifted: Player Mode Sketch

Indicating Action
Lifted: Completed Game
Diagramming Action

Diagram showing the concepts of Risk and Reward on the left, and Easy and Hard on the right. Arrows indicate the flow between these concepts.
Objectives

• Anything a player might strive for

• May be a **primary** game objective
  • Progressing the story
  • “Completing” the game

• May be an **auxiliary** game objective
  • Side missions/quests
  • Unusual achievements

• Sometimes **player-directed**
  • Reward structure in sandbox games
Objectives

- **Primary** objectives reflect vision
  - Wish fulfillment: *I want to ________*
  - Help player realize the dream

- **Auxiliary** objectives address player style
  - Achievements for *achievers*
  - Easter eggs for *explorers*
  - Online resources for *socializers*

- **Player-driven** objectives require a different focus
  - Start with a *toy*, and layer dramatic elements on it
Some Objective Categories

- **Capture**: take or destroy something of value
  - Includes “kill all enemies of type X”

- **Race**: reach a goal within time

- **Chase**: catch or elude an opponent
  - Race with a dynamic goal/destination

- **Rescue/Escape**: Get someone to safety

- **Exploration**: Locate something in game world
Some Objective Categories

- **Capture**: take or destroy something of value
  - Includes “kill all enemies of type X”

- **Race**: reach a goal within time

- **Chase**: catch or elude an opponent
  - Race with a dynamic goal/destination

- **Rescue/Escape**: Get someone to safety

- **Exploration**: Locate something in game world
Actions

- **Verbs** that describe what the player can do
  - Walk
  - Run
  - Jump
  - Shoot

- **Does not need to be attached to an avatar**
  - Build
  - Swap
  - Rotate
**Actions**

- **Verbs** that describe what the player can **do**
  - Walk  
    (left or right)
  - Run  
    (walk, but faster!)
  - Jump  
    (up; jump/run for left or right)
  - Shoot  
    (left or right)

- Does not need to be attached to an avatar
  - Build
  - Swap
  - Rotate
Actions

• **Verbs** that describe what the player can do
  • Walk (left or right)
  • Run (walk, but faster!)
  • Jump (up; jump/run for left or right)
  • Shoot (left or right)

• Does not need to be attached to an avatar
  • Build
  • Swap
  • Rotate

**Action Platformer**
**Actions**

- **Verbs** that describe what the player can **do**
  - Walk (left or right)
  - Run (walk, but faster!)
  - Jump (up; jump/run for left or right)
  - Shoot (left or right)

- **Does not need to be attached to an avatar**
  - Build (RTS or simulation)
  - Swap (Bejeweled clones)
  - Rotate (Stacking games)
Designing Actions

- Starts with brainstorming the verbs
  - Define the types of verbs
  - Define the scope of the verbs

- Design Goals
  - Enough verbs to avoid being too simple
  - But not so much to be confusing (verb bloat)
  - Do the verbs directly achieve the goal?

- Each verb maps to a single input

Primary Actions

- How do verbs, goals relate?
  - Imagine there no challenges
  - What verbs *must* you have?

- **Example**: Platformers
  - **Goal**: reach exit location
  - Only need movement verbs
  - Killing enemies is *optional*
  - Other actions are *secondary*

- **Focus on primary actions**
Some Secondary Actions are Fine

- Often in **puzzle platformers**
  - Platformer verbs + something
  - “Innovation on the cheap”

- Verb alters “geography”
  - Access hard-to-reach areas
  - Directly overcome *challenges*
  - Really just movement+

- But do this sparingly!
  - Too many creates **verb bloat**
The Game State

- Collection of values representing game world
  - Location, physical attributes of each game object
  - Non-spatial values (e.g. health) of these objects
  - Global non-spatial values (e.g. difficulty)

- Actions *modify* the game state

- Not necessary to specify this in early designs
  - Focus on coming up with your actions first
  - Only need enough state to understand *interactions*
Interactions

- Not a *direct* action of player
  - Result of the *game state*
  - Can happen w/o controller

- **Example**: collisions
  - May be bad (*take damage*)
  - May be good (*power-up*)

- **Other Examples**:
  - Spatial proximity
  - Line-of-sight
  - Resource acquisition
Game Mechanics

- **Game mechanic**
  - Relationship of **verbs**, **interactions**, and **state**
  - Often call this relationship the “rules”
  - **Gameplay** is manifestation of these rules

- **Example**: Joust
  - **Verbs**: Flap; go left or right
  - **Interaction**: Collision with opponent
  - **Rule**: If hit opponent, lower player dies
Gameplay Example: *Joust*
Verbs vs Interactions

- **Design Idea**: minimalism
  - Game with very few verbs
  - Mechanics are all interactions
  - Common in mobile, tablet

- **Example**: Sneak Beat Bandit
  - Has only one verb: *move*
  - Rhythm game; move to beat
  - All movement on rails
  - If obstacle in way, turn
  - Line-of-sight mechanics
Beat Sneak Bandit
Avoid Verb Proxies

**Proxy**: verb that activates another verb
- “Use an item” (what does the item do?)
- “Shoot” (what does the weapon do?)

**Make your verbs outcome oriented**
- Fire standard projectile (like shoot, but says what it shoots)
- Fire freezing beam (what is does and how it is applied)

**Important questions to ask**
- Does it help me reach a goal?
- Does it overcome a challenge?
Avoid Verb Proxies

- **Proxy**: verb that activates another verb
  - “Use an item” (what does the item do?)
  - “Shoot” (what does the weapon do?)

- Make your verbs **outcome oriented**
  - Fire **standard projectile** (like shoot, but says what it shoots)
  - Fire freezing beam (what it does and how it is applied)

- Important questions to ask
  - Does it help me reach a goal?
  - Does it overcome a challenge?
Challenges

• **Obstacles**
  • Prevent progress towards goal
  • Have to be “overcome”

• **Opponents**
  • Players or bots with their own goals
  • May or may not need to be overcome

• **Dilemmas**
  • Can only perform one of several actions
  • “Correct” choice not immediately clear
Challenges

- **Obstacles**
  - Prevent progress towards goal
  - Have to be “overcome”

- **Opponents**
  - Players or bots with their own goals
  - May or may not need to be overcome

- **Dilemmas**
  - Can only perform one of several actions
  - “Correct” choice not immediately clear

See Text for Specific Examples
Challenges: Limitations

- **You cannot** always perform an action
  - Shooting may require ammo
  - Cannot (always) jump in mid air

- **Limitation**: requirement to perform action
  - Boolean test (like an \textit{if--then})
  - Checked at time of user input

- Only **one** limitation per verb
  - If more than one, split into more verbs
  - Reason double-jump is distinct
Challenges: Limitations

- You **cannot** always perform an action
  - Shooting may require **ammo**
  - Cannot (always) jump in mid air

- **Limitation**: requirement to perform action
  - Boolean test (like an `if-then`)
  - Checked at time of user input

- **Only one** limitation per verb
  - If more than one, split into more verbs
  - Reason double-jump is distinct
Challenges: Resources

- Resources are **non-spatial** part of game state
  - Any value not a location or physical attribute
  - May be global or attached to an entity

- Examples
  - **Entity**: ammunition, health points
  - **Global**: enemy spawns, time remaining

- Resources often implement **limitations**

- They also define the **game economy**
Challenges: Resources

- Resources are **non-spatial** part of game state
  - Any value not a location or physical attribute
  - May be global or attached to an entity

- Examples
  - Entity: ammunition, health points
  - Global: enemy spawns, time remaining

- Resources often implement **limitations**

- They also define the **game economy**

Will cover in more detail later.
Putting It All Together

• Start with your **vision**
  • I want to __________
  • This creates setting and player goals

• Create a (partial) list of the following:
  • **Objectives**
  • **Actions**
  • **Interactions**
  • **Challenges**

  Sketch **player modes** to show them in action