Lecture 2

Mechanics Revisited
Announcement: Visual Studio

• **Solution 1:** Multiple IDEs
  • Develop and test on Windows with Visual Studio
  • Load and test on Android with Eclipse
  • John Oliver has done this; **install party soon**

• **Solution 2:** Bleeding Edge
  • You must use VS 2015 (Preview)
  • You must have Hyper-V (in Windows Pro)
  • Both are available from Dreamspark
Purpose of Today’s Lecture

- Give a review of formal design elements
  - Not everyone here has had the Intro Games course
  - And for the rest of you, it has been over a year

- Develop a deeper understanding of mechanics
  - Understand the important of interactions
  - Understand the analysis challenges

- Set us up for the later lectures on mechanics
  - Monetization and mobile game design
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)
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*

• **Goal**: Focus on primary
  • Secondary verbs lead to bloat
  • Add features with interactions
Secondary Actions are Acceptable

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

- Verb that alters “geography”
  - Access hard-to-reach areas
  - Directly overcome *challenges*
  - Not directly needed for goal

- But do this sparingly!
  - Indies have one new verb!
  - Other features are *interactions*
Interactions

- Not a *direct* action of player
  - Outcome of the *game state*
  - Can happen without controller

- **Example**: collisions
  - Accidental or player forced
  - May be bad (*take damage*)
  - May be good (*gain power-up*)

- **Other Examples**:
  - Spatial proximity
  - Line-of-sight
  - Resource acquisition
• **Game mechanic**
  • Relationship between verbs and interactions
  • 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*
Design Goal: Verb Minimalism

- Can we limit to one verb?
  - Mechanics are all interactions
  - Common in mobile, tablet
  - Due to lack of input modes

- **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 it 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”
  • “Shoot”

• 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?

Behavior is defined by *interaction* of projectile with the environment.
Understanding Game State

- Many game state values are **spatial**
  - Represent location of a game **entity**
  - Also physical values like velocity, acceleration

- Entities act as containers for non-spatial values
  - Values that never change: **attributes**
  - Values that can change: **resources**

- Attributes, resources can be global as well
  - Though most mechanics are at entity level…
Actions Affecting Spatial State

• Typically we what we would call *movement*

• But there are many ways to implement
  • **Direct** movement of avatar (e.g. WASD)
  • **Indirect** movement of avatar (e.g. pathfinding)
  • Alter the **environment** (e.g. removing platforms)

• Area of much potential *innovation*
  • Particularly given the limitations of mobile
Altering the Environment

- Found in “physics” games
  - No direct control of avatar
  - Can only remove/add/move obstacles in environment
  - Movement is “natural”

- **Example**: *Screw the Nut*

- Physics is a rule system
  - Interaction, not action
  - Takes one state to another
  - Also one that is complex to understand/model
Innovating Avatar Movement

- 2D games move on 2-axes
  - Classic: left-right/up-down
  - Unless top-down game, one of these axes is restricted
- Is jump the only option?
  - Launcher/trajectory verbs
  - (Limited) teleportation
- **Example**: *Knightmare Tower*
  - Launcher-style game
  - Vertical movement is boosts gained from killing enemies
Environment **AND** Avatar

- Possible to split the verbs
  - Some for avatar movement
  - Others for environment
- Found in “drawing” games
  - Draw missing platforms
  - Avatar walks on platforms
  - **Ex:** Max & Magic Marker
- Innovate by limiting avatar
  - Move on single axis
  - Combine with environment
  - **Example:** Swindler
Combining Actions

- Verbs can combine in interesting ways
  - Run and jump in a platformer
  - Strafing fire in a shooter

- Typically result of the interactions
  - Each verb interacts with environment in different way
  - Combination of two give extra feature for “free”
  - This is an example of emergent behavior

- Not all combinations are emergent
  - **Example**: Double jump is not a feature of interactions
  - This type of verb combination is a distinct action
# Combining Actions

## Running Jump
- Can move while in midair
  - Just horizontal movement
  - Not realistic; it is a game
  - Many platformer challenges assume this type of control
- Different than a *long jump*
  - Less height than reg. jump
  - No control once in the air
  - Would be a **distinct action**

## Strafing Fire
- Based on “real life” property
  - Bullets travel in straight line
  - Movement changes origin
  - Walking side-side makes a spray (used in covering fire)
- But some features are gamy
  - Bullets slower than life
  - Character faster than life
  - Creates interesting effects
Combining Actions

**Jump**
- Can move while in midair
  - Just horizontal movement
  - Not realistic; it is a game
  - Many platformer challenges assume this type of control
- Different than a *long jump*
  - Less height than reg. jump
  - No control once in the air
  - Would be a **distinct action**

**Strafing Fire**
- Based on “real life” property
  - Bullets travel in straight line
  - Movement changes origin
  - Walking side-side makes a spray (used in covering fire)
- But some features are gamy
  - Bullets slower than life
  - Character faster than life
  - Creates interesting effects

**Interaction(?)**

**Interaction**
Combining Actions

Is this an example? **Why** or **why not**?
## Common Spatial Interactions

### Collisions
- Can effect *resources*
  - Player takes damage
  - Player gains power-up
  - Player-NPC transfer gold
- Can effect *spatial values*
  - Bounce off collision point
  - Swing from attached rope
  - Attraction to magnet/charge

### Detection
- Examples:
  - Line-of-sight (w/ obstacles)
  - Spatial proximity
- Can have *direct* effects
  - Alarms in a stealth game
- Can have *indirect* effects
  - Tower defense targeting
  - Adjust NPC reactions
## Resource-Spatial Interactions

### Resource Affects Spatial

- Resources can unlock areas
  - Keys are a trivial resource
  - Also use resource thresholds
  - **Ex:** Collect all tokens to pass

- Resources affect difficulty
  - Adjust input device sensitivity
  - **Ex:** Deadeye meter in *RDR*
  - **Ex:** Jet packs to increase jump

### Spatial Affects Resources

- Resources made by entities
  - Have a spatial location
  - **Ex:** Time to transfer resources
  - **Ex:** Sources be captured

- Resource values are entities
  - Take up physical volume
  - Need space to acquire
  - **Ex:** Inventory in *Deux Ex*
Actions and 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)
  • **Example**: double jump is different from jump

• Primary use of resources in game design
  • Presence of resource allows action; may consume
Balancing Resources

- **Sources**: How a resource can increase
  - **Examples (player)**: ammunition clips, health packs
  - **Example (external)**: spawn points

- **Drains**: How a resource can decrease
  - **Examples (player)**: firing weapon, player damage
  - **Examples (external)**: monster death

- Adjust sources and sinks to “balance” economy
  - Together, determine “price” of resource
  - Price of resource should reflect its “power”
Design Problem: Pricing Resources

Underpricing

• Cheap, powerful actions
  • Players favor these verbs
  • Limits play variety

• Examples:
  • Buff spells in most RPGs
  • *Dragon Age* cold spells
Design Problem: Pricing Resources

Overpricing

- Expensive, weak actions
- Usage is “penalized”
- Waste of designers’ time

Examples:
- Shredder ammo in ME2
- *Raise Dead* in early D&D
## Design Problem: Pricing Resources

### Underpricing
- Cheap, powerful actions
  - Players favor these verbs
  - Limits play variety
- Examples:
  - Buff spells in most RPGs
  - *Dragon Age* cold spells

### Overpricing
- Expensive, weak actions
  - Usage is “penalized”
  - Waste of designers’ time
- Examples:
  - Shredder ammo in ME2
  - *Raise Dead* in early D&D

- Resource usage determines difficulty
  - *Resident Evil*: Availability of ammunition
  - D&D 3.x: 20% resource per encounter
Resource Analysis: Dungelot

- Simple combat mechanic
  - Each round, swap damage
  - Enemy dies when health is 0
- Player goes until health is 0
  - There is healing in game
  - …but too sparse to go forever
- Two primary characters
  - **Paladin**: can lessen damage
  - **Vampire**: drains blood to heal
  - Which is better?
Bad Design: “Engines”

- Actions combine to make resources free
  - Spend one resource to get another
  - Use new resource to get old one back

- **Example:** *Dragon Age: Origins*
  - Resources: Health, Mana
  - Small health loss; regain much mana
  - Small mana loss; heal much damage

- **Solution?** Cool-down time
Summary

- **Mechanics** combine **actions** and **interactions**
  - Actions are a direct result of player controls
  - Interactions triggered by a particular game state
  - Input limitations make interactions very important

- Interactions depend on the **game state**
  - Spatial state associated with physics, detection
  - Resources associated with limitations, unlocking