SE2250 – Software Construction

Lecture #8

Requirements, Analysis and Design

March 18, 2019

Outline

- Development activities in your project – phase 3
- Development activities – Apple Picker example
- Design Goals
- Guiding the Player
- Game balance

Development activities

- **Project – phase 3**
  - ***Requirements elicitation*** ***(A1)*** results in the specification of the system that the client understands
  - Software Requirements Specification (SRS) is a description of a software system to be developed (intended for software users)
  - Functional and non-functional requirements
  - May include a set of use cases that describe user interactions that the software must provide

Development activities

- ***Analysis (A2)*** results in an analysis model that the developers can unambiguously interpret (class diagrams, sequence diagrams, state machines…)

- Models?
  - **functional model (A1)**
    - represented by use cases and scenarios
  - **analysis object model (A2)**
    - represented by class and object diagrams
  - **dynamic model (A2)**
    - represented by state machine and sequence diagrams

Development activities

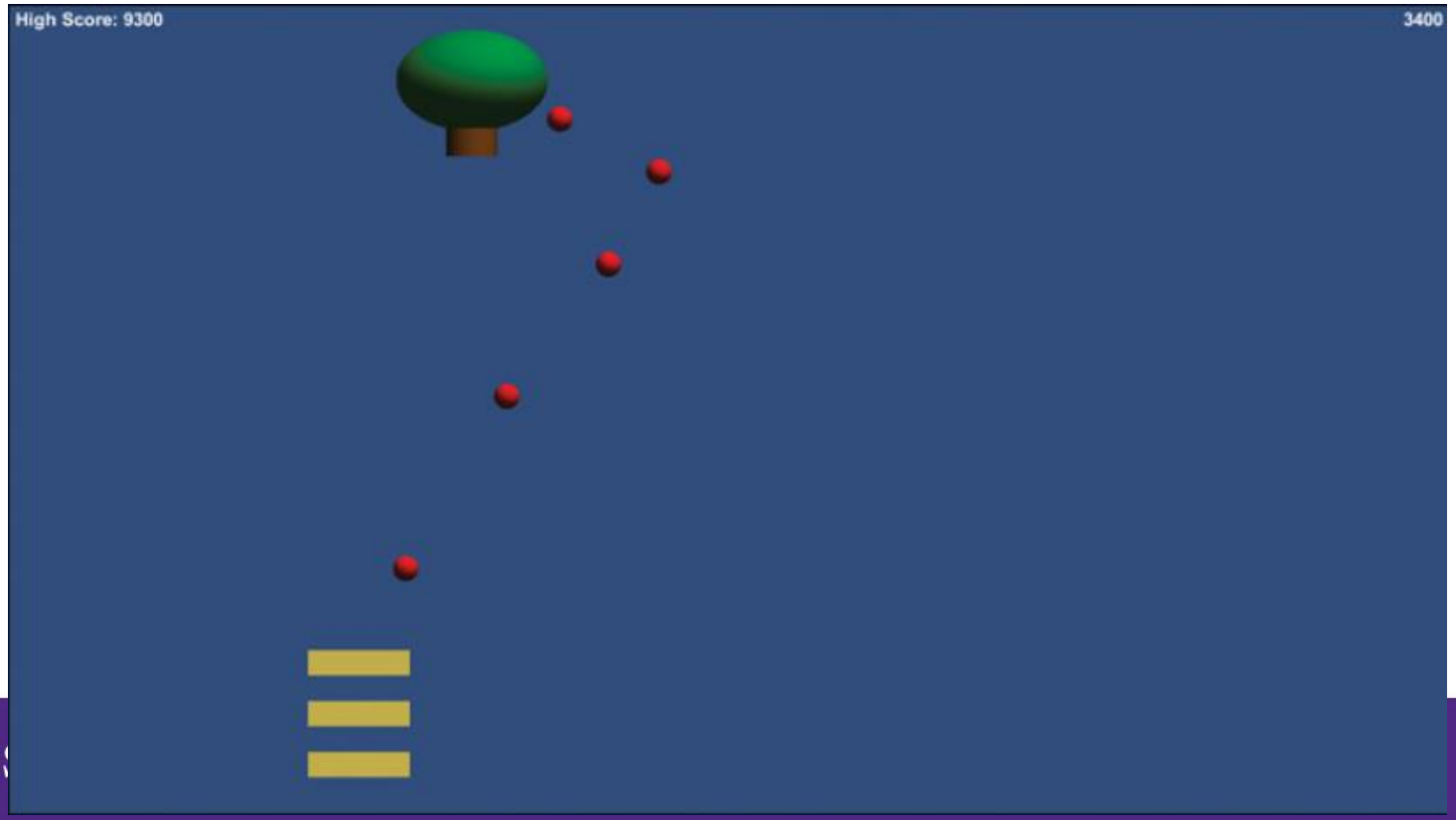
- **System design (A3)**
  - ***design goals***, describing the qualities of the system to optimize
    - Derived from non-functional requirements
    - Trade-offs are commonly needed
  - ***boundary*** *use cases*
    - Configuration
    - Start-up and shutdown
    - Exception handling

Development activities

- **Object Design (A4)**
- Includes:
  - ***Reuse* -** off-the-shelf components, class libraries, design patterns are adapted
  - ***Interface specification* -** a complete interface specification for each subsystem
  - ***Restructuring*** – manipulate the system to increase reuse
  - ***Optimization* -** address performance requirements

Apple Picker

- **Apple Picker**
- The player moves baskets back and forth in an attempt to collect apples that are falling from a tree



Apple Picker

- **Apple Picker - requirements**
  - The player controls the three baskets at the bottom of the screen and can move them left and right using the mouse.
  - The apple tree moves left and right randomly while dropping apples.
  - The player must use her/his baskets to catch the apples before they hit the ground
  - For each apple that the player catches, (s)he earns points
  - If even a single apple hits the ground, it and all other remaining apples disappear, and the player loses a basket
  - When the player loses all three baskets, the game is over

Apple Picker

- **Apple Picker**
- Analysis and design:
  - Identify objects in Apple Picker (mostly GameObjects)
  - Analyze each of their behaviours
  - Break down those behaviors to simple commands
  - Show behaviours in flowcharts (you can use other UML in your assignment)

Apple Picker

- **GameObjects in Apple Picker:**
  - **Baskets:** Controlled by the player, the three Baskets move left and right following the player's mouse movements. When a Basket collides with an Apple, the Apple is caught, and the player gains points.
  - **Apples:** The Apples are dropped by the AppleTree and fall straight down. If an Apple collides with the Baskets, the Apple is caught and disappears from the screen. If an Apple passes off the bottom of the screen, it disappears, and it causes all other Apples to disappear. This destroys the top Basket.
  - **AppleTree:** The AppleTree moves left and right randomly while dropping Apples. The Apples are dropped at a regular interval, so the only randomness in the behavior is the left and right movement.

Apple Picker

- **Basket - actions**
  - Move left and right following the player's mouse.
  - If any Basket collides with an Apple, catch the Apple and award points

  - The behaviour of the Basket in a flowchart
  - The game loops through this flowchart every frame (*oval*)
  - *Boxes* – actions
  - *Diamonds* - decisions

Apple Picker

- **Basket flowchart**

Apple Picker

- **Apple - actions**
  - Fall down.
  - If an Apple hits the bottom of the screen, the end of the round is triggered

  - *Note*: The collision between the Apple and the Basket is part of the Basket behavior, so it does not need to be handled in the Apple flowchart

Apple Picker

- **Apple flowchart**

Apple Picker

- **AppleTree - actions**
  - Move left and right randomly
  - Drop an Apple every 0.5 seconds

  - Two decisions to make each frame
    - Does it change direction?
    - Does it drop an Apple?

Apple Picker

- **AppleTree flowchart**

Design Goals

- **Design Goals**
- Many possible goals when designing a game or interactive experience
  - **Designer-centric goals**
    - Focus on you as the designer
    - What do you personally want to get out of designing this game?
  - **Player-centric goals**
    - Focus on the players
    - What do you want for the players of your game?

Design Goals

- **Designer-centric goals**
  - **Fortune**: You want to make money
  - **Fame**: You want people to know who you are
  - **Community**: You want to be part of something
  - **Personal expression**: You want to communicate with others through games
  - **Greater good**: You want to make the world better in some way
  - **Becoming a better designer**: You simply want to make games and improve your craft
  - …

Design Goals

- **Player-Centric goals**
  - **Fun**: You want players to enjoy your game.
  - **Lusory attitude**: You want players to take part in the fantasy of your game.
  - **Flow**: You want players to be optimally challenged.
  - **Structured conflict**: You want to give players a way to combat others or challenge your game systems.
  - **Empowerment**: You want players to feel powerful.

Design Goals

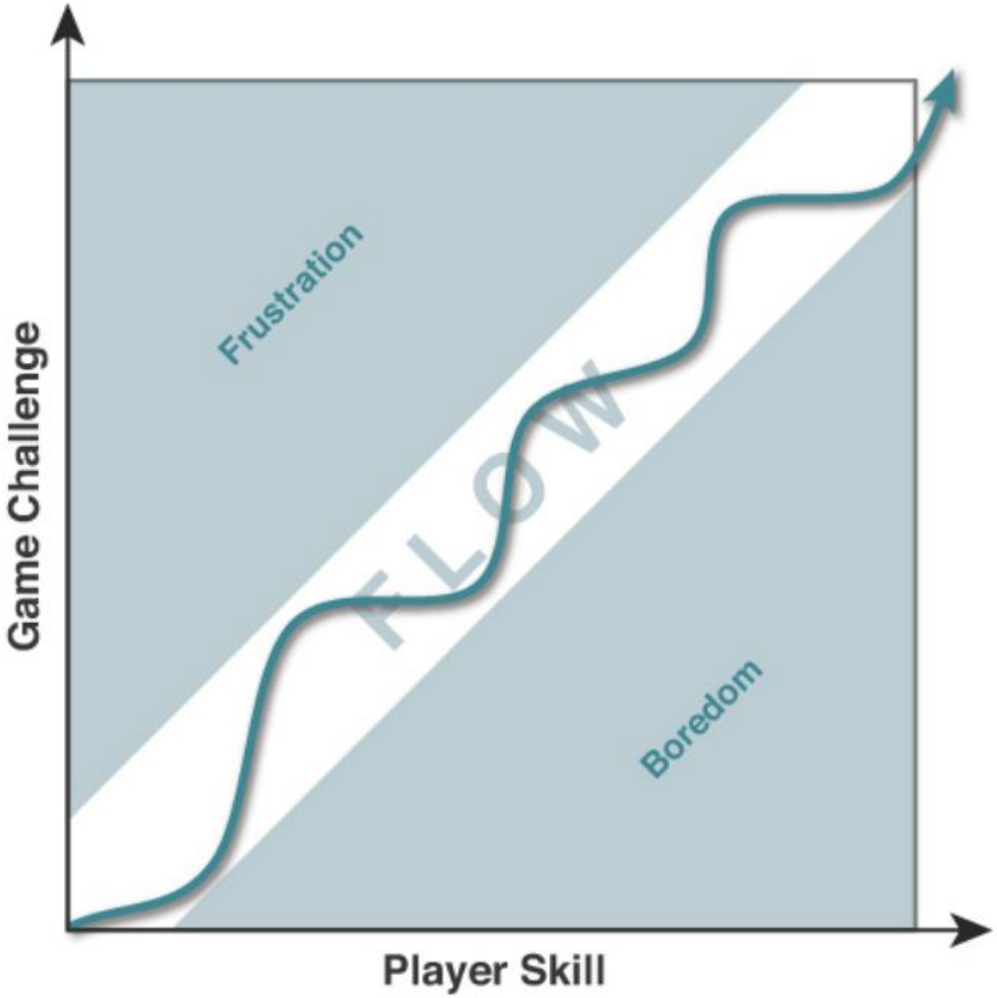
- **Player-Centric goals…continued**
  - **Interest / attention / involvement**: You want players to be engaged by your game.
  - **Meaningful decisions**: You want players' choices to have meaning to them and the game.
  - **Experiential understanding**: You want players to gain understanding through play.
  - ….

Design Goals

- **Flow**
  - Flow is the state of optimal challenge.
  - In a flow state, a player is focused intently on the challenge before her/him and often loses awareness of things that are outside of the flow experience.
  - The flow state exists between boredom and frustration
  - If the game is too challenging for the player's skill level, (s)he will feel frustrated.
  - Conversely, if the player is too skilled for the game, (s)he will feel bored.

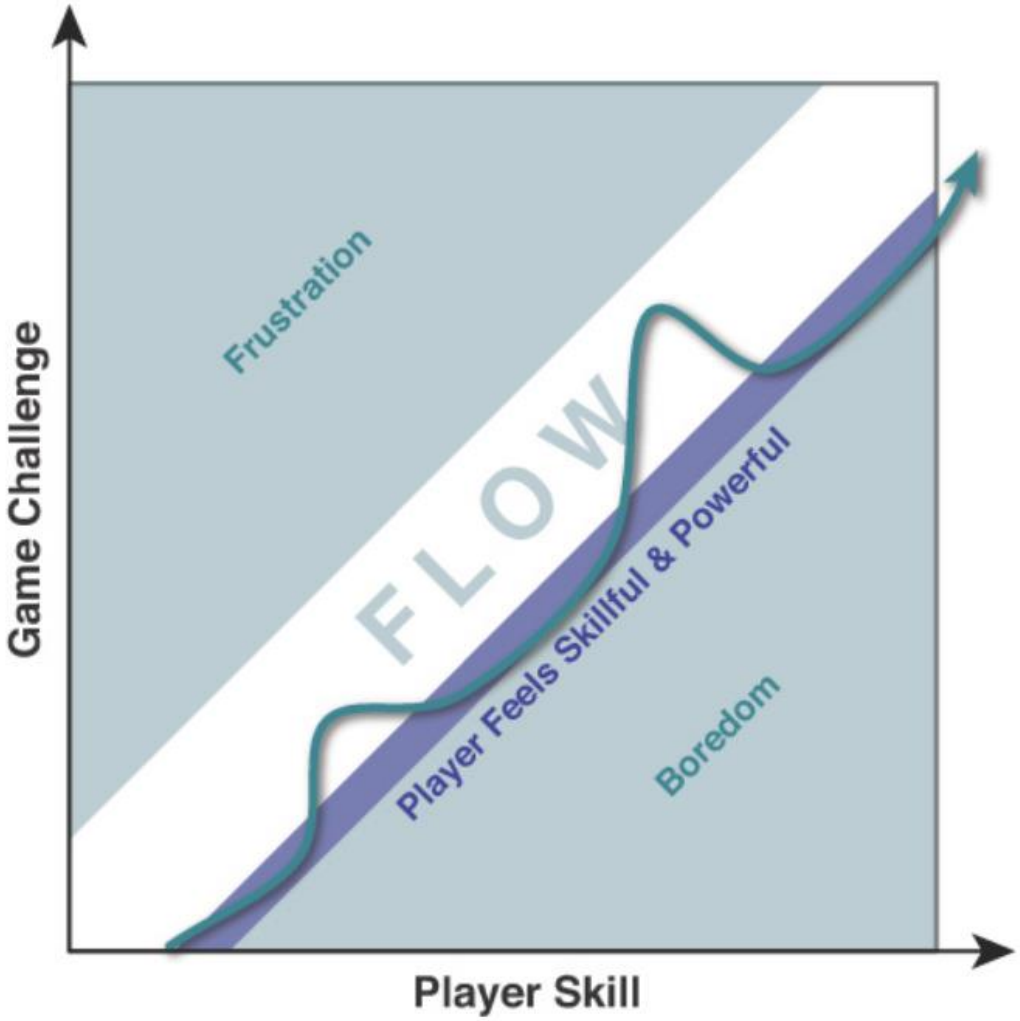
Design Goals

- **Flow**



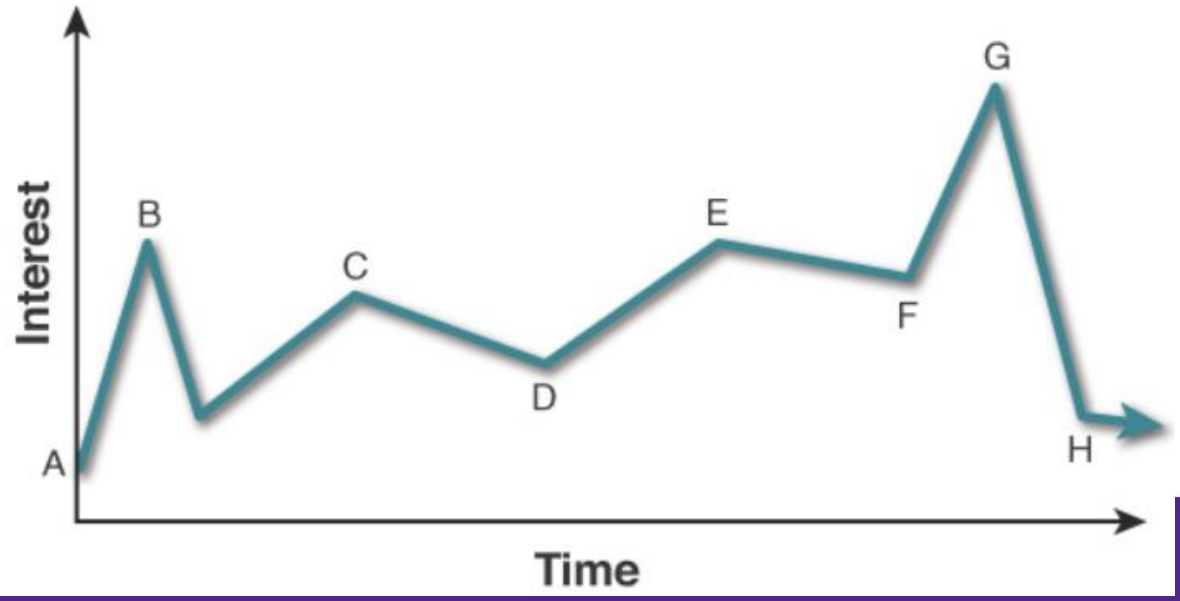
Design Goals

- **Flow**



Design Goals

- **Attention and Involvement**
  - enter - a little interest (A)
  - grab with a "hook" that piques interest (B)
  - can drop it back down and steadily build interest with little peaks and valleys (C, E and D, F)
  - build to the highest (G)
  - close (H)



Guiding the Player

- A designer primary job is to craft an experience for players to enjoy
- Make sure that players who have never seen the game before also intuitively understanding what they need to do to experience the game as you intended

Guiding the Player

- **Guiding the player - Two styles:**
  - **Direct** - the player knows that (s)he is being guided
  - **Indirect** - the guidance is so subtle that players often don't even realize that the guidance is there

Guiding the Player

- **Direct guidance**
- Quality is determined by:
  - *Immediacy* - message must be given to the player when it is immediately relevant
  - *Scarcity* – scarce messages are more valuable, more likely to be heard/followed
  - *Brevity* – as short as possible, only what necessary
    - "When near sandbags, press O to take cover and reduce damage from enemy attacks."
  - *Clarity* – provide sufficient and clear instructions
    - "When standing near sandbags, press O to take cover"

Guiding the Player

- **Direct guidance**
- *Instructions*
  - The game explicitly tells the player what to do
- *Call to Action*
  - The game explicitly gives the player an action to perform and a reason to do so (missions)
- *Map or Guidance System*
  - A map directs the player toward the goals
- *Pop-Ups*
  - Contextual controls that change based on the objects near the player

Guiding the Player

- **Indirect guidance**
  - The art of influencing and guiding the player without her/him actually knowing that she/he is being controlled
- Quality attributes:
  - **Invisibility**: How aware is the player that she is being guided? Will her awareness negatively impact her experience of the game?
  - **Reliability**: How often does the indirect guidance influence the player to do what you want?
    - Most players in a dark area of the game will head toward a lit doorway, but not all

Guiding the Player

- **Indirect guidance**
- *Constraints*
  - give the player limited choices. Too many choices - choice paralysis
- *Goals*
  - if the player has a goal to collect bananas and has two possible doors to go through, placing clearly visible bananas behind one of the doors will guide the player
- *Physical Interface*
  - If you give a player of Guitar Hero or Rock Band a guitar-shaped controller, (s)he will generally expect to use it to play music

Guiding the Player

- **Indirect guidance**
- Visual design
  - Light, similarity, trails, landmarks, arrows…
- Audio design
  - Can be used to influence mood and behaviour
- Player Avatar
  - If the player character looks like a rock star holding a guitar, the player might expect for her character to be able to play music.
- Non-player characters
  - Modeling behaviour (positive, negative, safety)

Game balance

- Balance means different things depending on the context
- Multiplayer game
  - Fairness - each player should have an equal chance of winning the game
    - Symmetric games - each player has the same starting point and abilities
- Single-player game
  - Appropriate level of difficulty for the player and the difficulty changes gradually

Game balance

- Parts of game balance
  - Probabilities
  - Randomizers
  - Distributions/weighted distributions
  - Permutations
  - Positive and negative feedback

Game balance

- Probabilities
  - Probabilities range from 0 to 1
  - Probability is "sought outcomes" divided by "possible outcomes"
  - Enumeration
  - When sought outcomes are mutually exclusive, add their probabilities
  - When sought outcomes are not mutually exclusive, multiply their probabilities

Game balance

- Balancing weapons or abilities
- Weapons has values:
  - The number of shots fired at a time
  - The damage done by each shot
  - The chance that each shot will hit at a given distance
- Balancing weapons - want them to feel roughly equal to each other in power, though you also want each weapon to have a distinct personality

Game balance

- Balancing weapons

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	**Weapon**	**Shots**	**D/Shot**		**ToHit**									
2	**Original**				**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
3	Pistol	4	2		2	2	2	3	3	4	4	5	5	6
4	Rifle	3	3		4	3	2	2	2	3	3	3	4	4
5	Shotgun	1	10		2	2	3	3	4	5	6			
6	Sniper Rifle	1	8		6	5	4	4	3	3	2	2	3	4
7	Machine Gun	6	1		3	3	4	4	5	5	6	6		

- ToHit - the minimum roll on 1d6 that would hit at that range.
- Cell K3, the ToHit for the pistol at a range of 7 is 4, so if the player is shooting a target 7 spaces away, a roll of 4 or higher would be a hit

Game balance

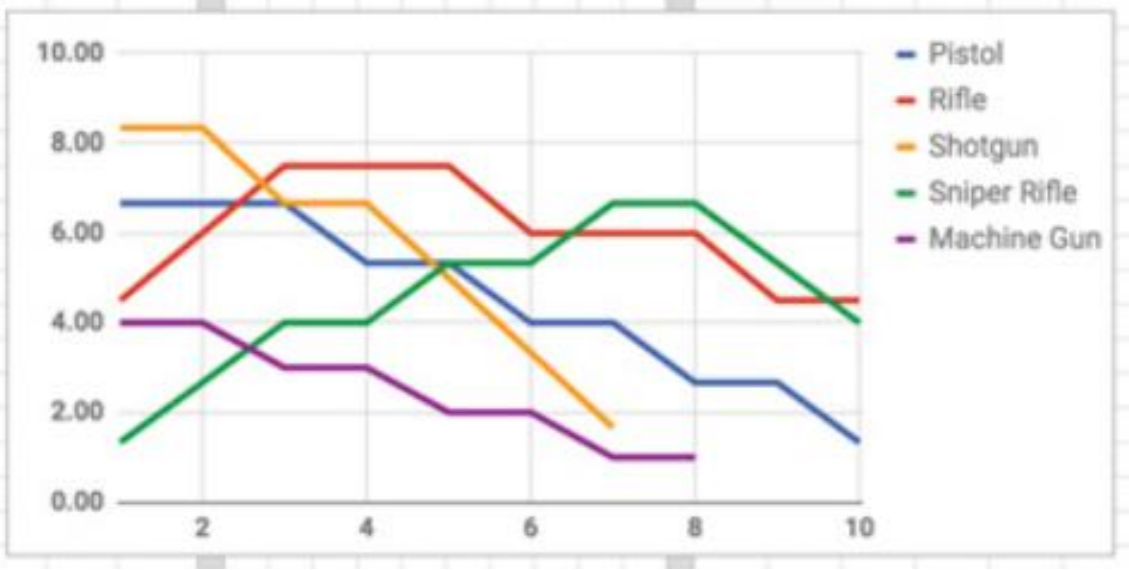
- Balancing weapons

	O	P	Q	R	S	T	U	V	W	X	Y
1		Percent Chance									
2		1	2	3	4	5	6	7	8	9	10
3		83%	83%	83%	67%	67%	50%	50%	33%	33%	17%
4		50%	67%	83%	83%	83%	67%	67%	67%	50%	50%
5		83%	83%	67%	67%	50%	33%	17%			
6		17%	33%	50%	50%	67%	67%	83%	83%	67%	50%
7		67%	67%	50%	50%	33%	33%	17%	17%		

AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK
Average Damage										
1	2	3	4	5	6	7	8	9	10	
6.67	6.67	6.67	5.33	5.33	4.00	4.00	2.67	2.67	1.33	
4.50	6.00	7.50	7.50	7.50	6.00	6.00	6.00	4.50	4.50	
8.33	8.33	6.67	6.67	5.00	3.33	1.67				
1.33	2.67	4.00	4.00	5.33	5.33	6.67	6.67	5.33	4.00	
4.00	4.00	3.00	3.00	2.00	2.00	1.00	1.00			

Game balance

- Balancing weapons



- Is this what you really wanted? TEST