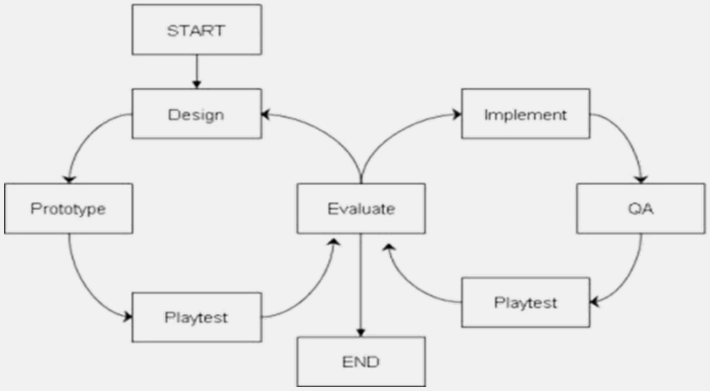
Creating a Game

Iterative Planning and Production

Rapid Prototyping and Mockups



Brent M. Dingle, Ph.D.
Game Design and Development Program
Mathematics, Statistics and Computer Science
University of Wisconsin - Stout

2015

See also references at end of slides (if any)

You Are More

- You are more than
  - an artist
  - a programmer
- You are studying to be
  - a Game Designer and Developer (GDDer)
- What does that mean?

A GDDer Understands

- All aspects of design and development
  - art
  - sound
  - programming
  - story presentation
  - workflow
  - management
  - advertising
  - communication
  - …

A GDDer Is

- **an artist**
- **a programmer**

- Pretty clear on these two

A GDDer Is

- an artist
- a programmer
- **an architect**

- Architects create blueprints
- Games require
  - Design Docs
  - Prototypes
- From blueprints
  - consistent (mass) production

A GDDer Is

- an artist
- a programmer
- an architect
- **a party host**

- Players are invited into the designer's "worlds"
  - for enjoyment and entertainment

A GDDer Is

- an artist
- a programmer
- an architect
- a party host
- **godlike**

- Creating worlds
- And the physical rules that govern those worlds

A GDDer Is

- an artist
- a programmer
- an architect
- a party host
- godlike
- **a lawyer**

- Create
  - and interpret
  - and explain
  - and defend/enforce
- rules that players must follow
  - and co-designers and developers must understand

A GDDer Is

- an artist
- a programmer
- an architect
- a party host
- godlike
- a lawyer
- **an educator**

- Many games require
  - presenting and teaching
    - new skills, ideas, concepts, frameworks of thinking
  - to players
    - and to co-designers/developers

A GDDer Is

- an artist
- a programmer
- an architect
- a party host
- godlike
- a lawyer
- an educator
- **a research scientist**

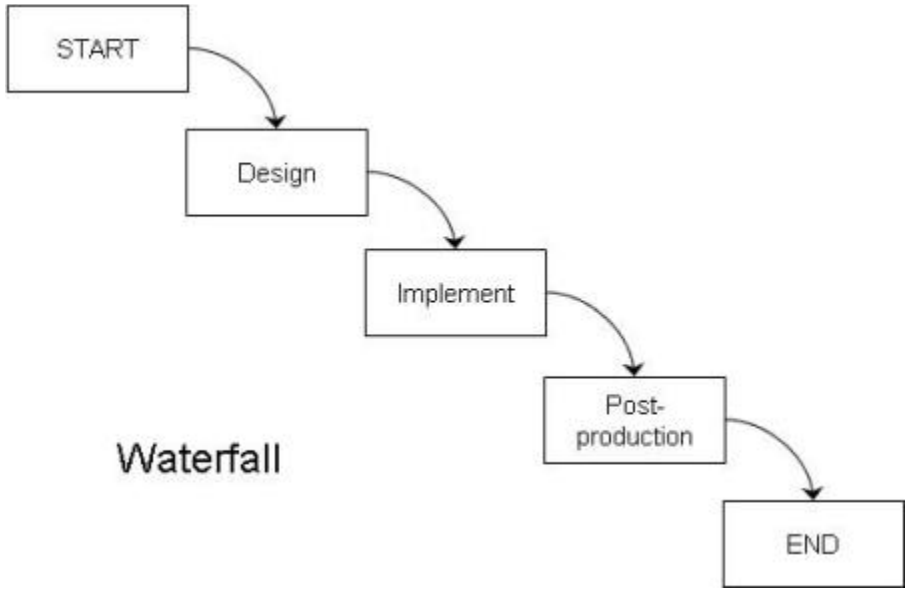
- Creating a game
  - Uses methods very much like the scientific method
  - Requires study and understanding of fields beyond ‘gaming, art, cs…’

Methods and Process

- Creating a game involves
  - planning and producing
  - both should be iterative and cyclic
- Why?
- Consider some history…

“Old” Process: Waterfall

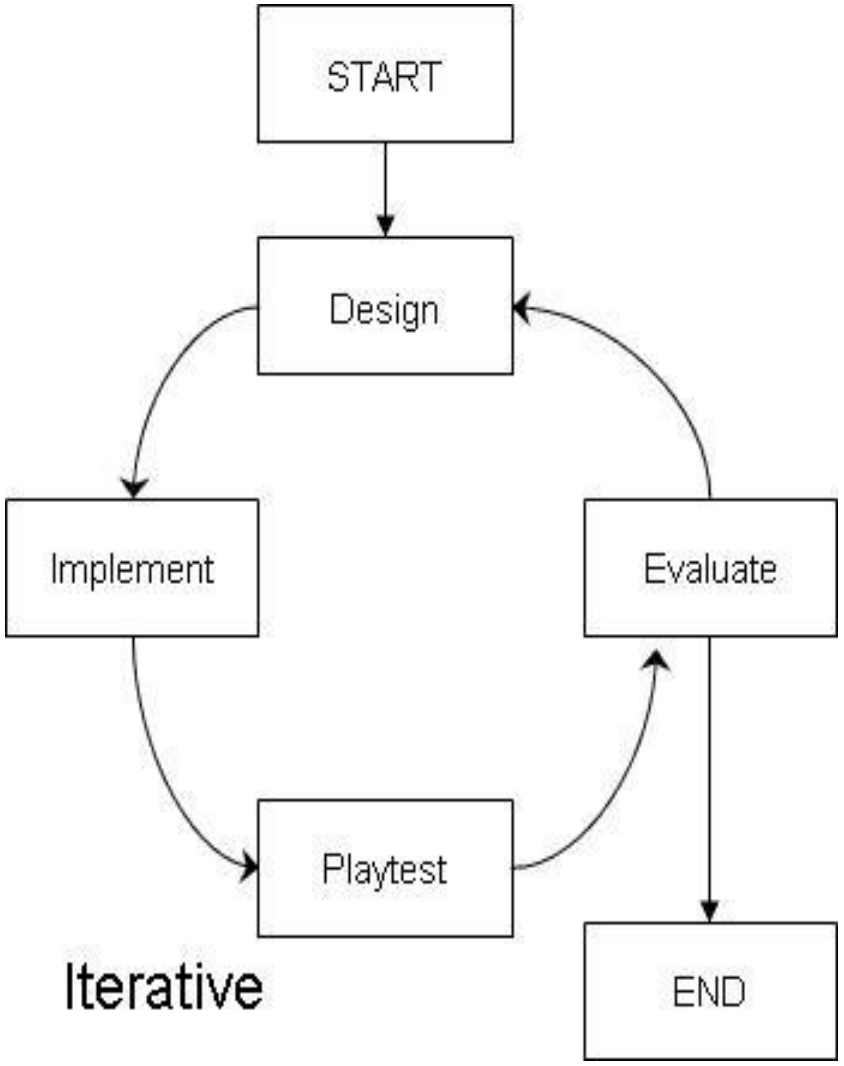
1. Design all on paper
2. Implement/program it
3. Test it
4. Add some “bling”
5. Ship it



Waterfall

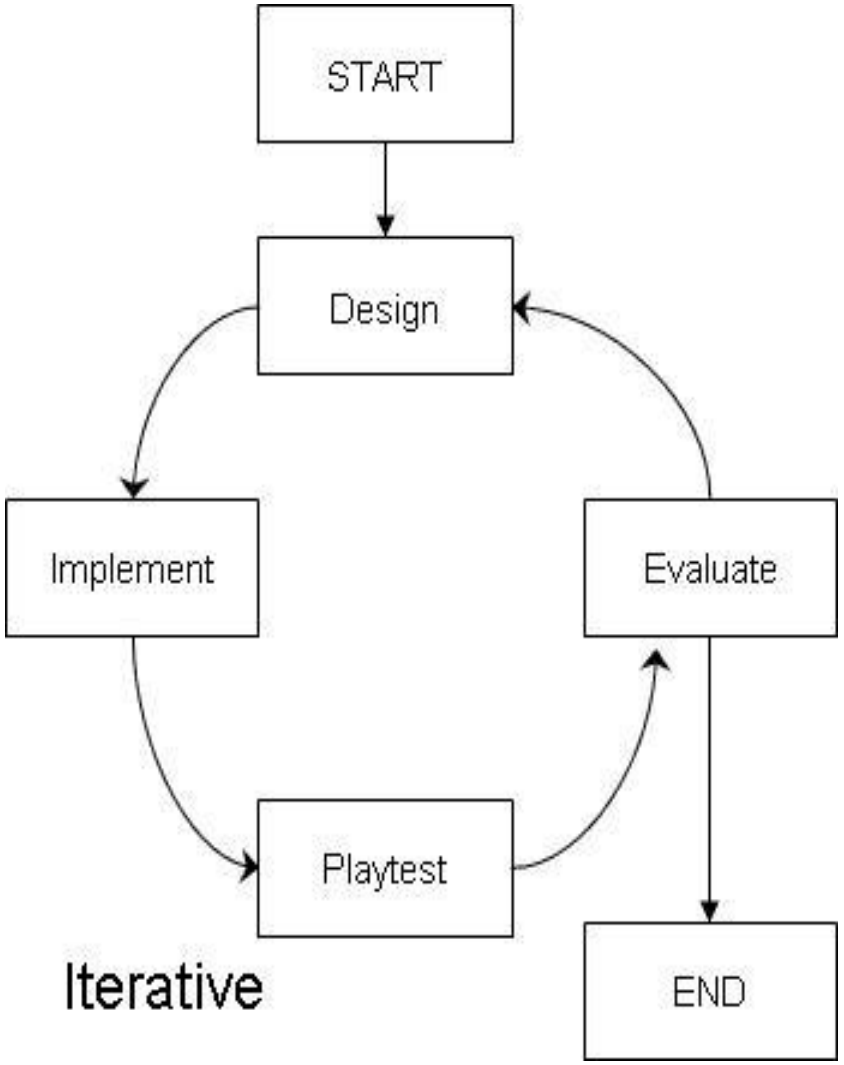
- *Assumes all things go perfect*
  - *Moves only in one direction*
  - *No way to go back and fix things*

“Newer” Processes are Iterative



- Start same as waterfall
  - Design
  - Implement
  - Test
- Add an extra phase
  - Evaluate
- Allows making changes if they are determined to be necessary

Link to Scientific Method



1. Make an observation
2. Propose hypothesis
3. Create Experiment to Test Hypothesis
4. Perform the Experiment
5. Evaluate Results
   - Repeat as necessary

More = Better

- The more iterations done
  - The better the game is likely to be

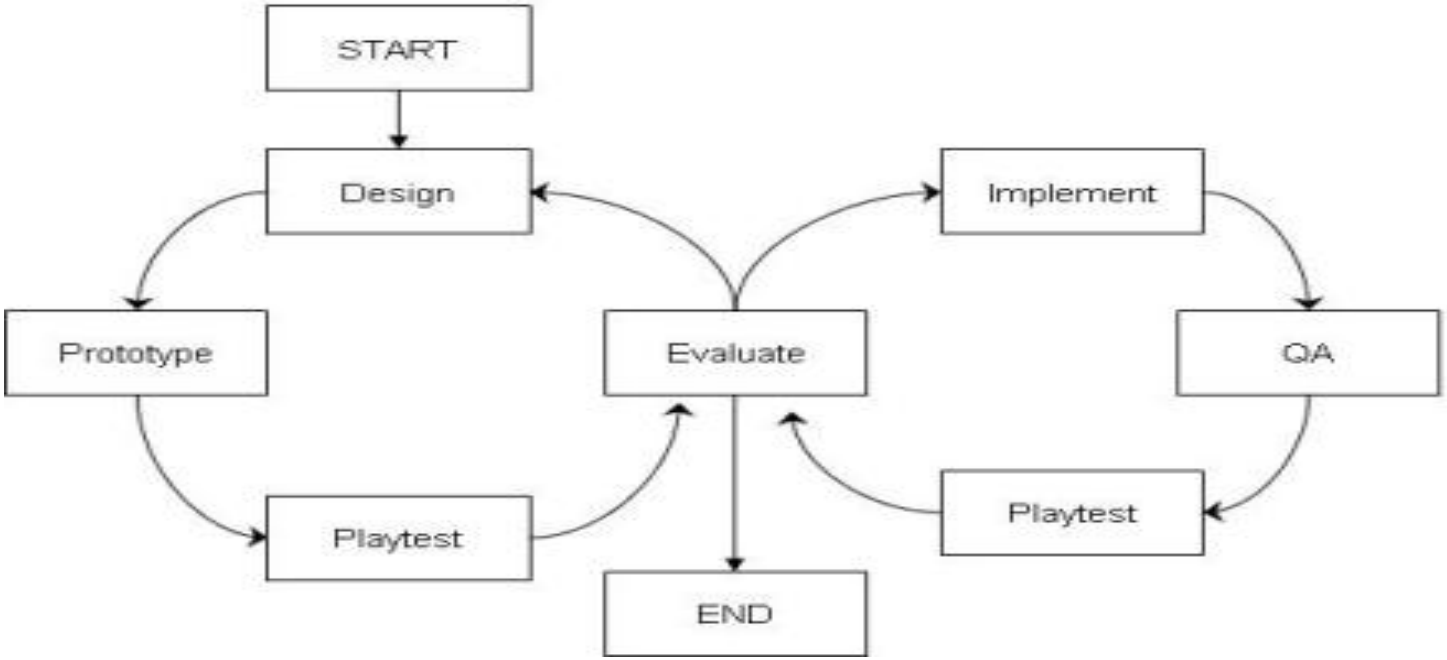
Snag on Video Games

- Implementation takes a LONG time
  - Programming is slow
    - Debugging is slower
  - Artwork can be slow
    - Finalizing is slower
  - Often difficult and time consuming to change once in place

Solution on Video Games

- Rapidly **Prototype** things
  - Usually on “paper only” first
  - Can be “easy digital”
    - i.e. little to no programming
- For clarity
  - for this class
  - we often/will call this a **mockup**

Iterative Rapid Prototyping



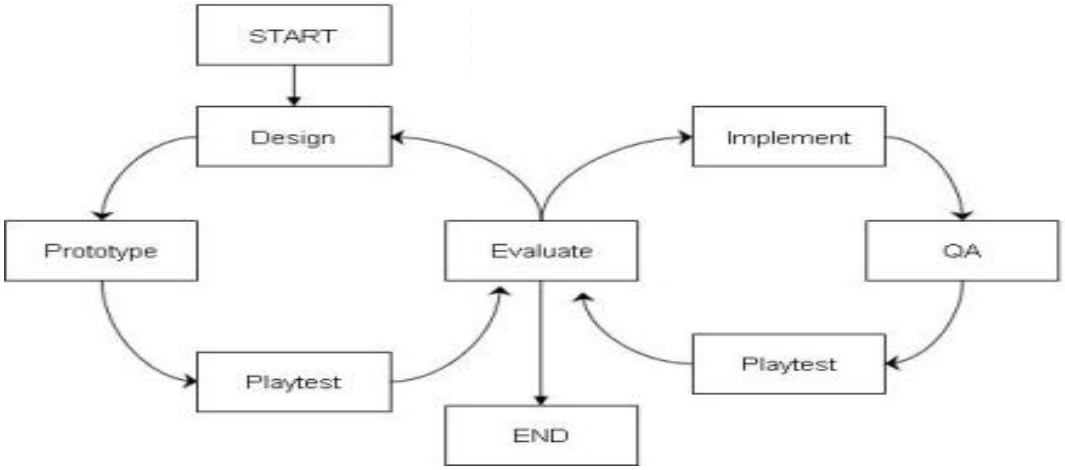
- More Iterations = Better Game
  - Iterate as many times as possible
  - Don’t start implementing until the design/planning is well done
    - Still possible to go back to design phase after implementation
    - But minimizes the likelihood

Risks

- Games have many risks
  - **Design Risk**
    - Risk game will not be fun and people won't like it
  - **Implementation Risk**
    - Possibility the development team will not be able to build the game
  - **Market Risk**
    - Game is great, but people don't buy it
  - …more

Risks Versus Rapid Prototyping

- Games have many risks
  - **Design Risk**
    - Risk game will not be fun and people won't like it



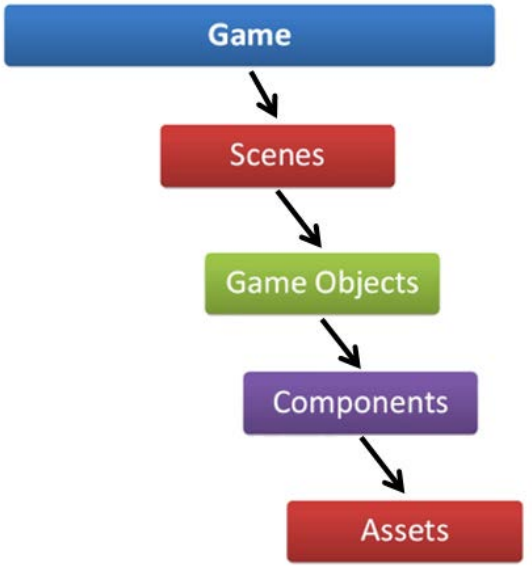
  - **Rapid Prototyping helps reduce Design Risk**

Risks Versus Game Decomposition

- Games have many risks

  - **Implementation Risk**
    - Possibility the development team will not be able to build the game

Later we will see how proper **Game Decomposition can reduce the Implementation Risk**



Also In the Future: Agile Design

- Later
  - We will take a closer look at **agile design**
    - And how **Rapid Prototyping relates**

For Now: Take Away

- The **greater the Design Risk**
  - *the **more unknowns** you have*
  - *the **more untested** and **unproven rules** your have*
- The **more iterations** you will **need**

Risk to Video Games

- Video games can have **impressive technology**
  - physics
  - graphics
  - sounds
  - …
- **Obscuring the "staleness"** of the gameplay

Suggestion/Requirement

- **Take the time to create mockups**
  - Do the iterations
  - Make the planning as solid as it can be

- The catch
  - Must **do so quickly** enough to **have time to implement**

End Summary

- Great designers do not design great games
  - Often the design is bad at the beginning
  - iteration and testing makes them become great

- Notables
  - Have a playable mockup of your game as early in development as possible
    - Faster this happens = faster you can test ideas
  - With equal amounts of time to create
    - Shorter, simpler game will be better than long & complicated
      - Simple Reason: Longer games take longer to iterate

Questions?

- Beyond D2L
  - Examples and information can be found online at:
    - http://docdingle.com/

    - *Continue to more stuff as needed*

References

- Some material in these slides was derived/based on material from:
  - Ian Schreiber, Game Design Concepts
  - https://gamedesignconcepts.wordpress.com/
  - Released under a Creative Commons Attribution 3.0 U.S. License
    - http://creativecommons.org/licenses/by/3.0/us/