PROJECT MANAGEMENT

PMU199: Intro to Game Design

Context for Game Design

- Game development is an agile development process.
  - Incremental development
    - Demonstrable product
    - Product milestones
  - Small groups
  - Changing requirements
  - Unexpected developments

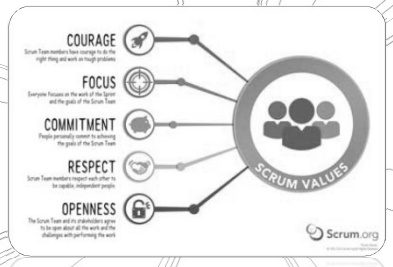
Recommended approach

- Constant communication
- Working build
- Shared workspace
- Team development
- Prioritize features
- Playtesting Playtesting Playtesting!

Biggest pitfalls

- Lack of communication
  - Diffusion of responsibility
- Lack of planning
  - Last-minute development
  - The curse of demo day
- Navel gazing
  - (Omphaloskepsis?)
- Feature management

Agile Development

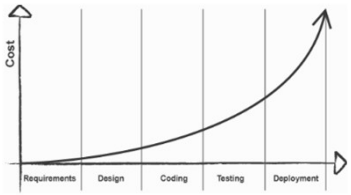


Sturdy Methodologies

- Also known as "traditional" or "waterfall" methods).
  - Underlying philosophy: Measure twice, cut once.
    - Or: The cheapest bug to fix is one that doesn't exist.
  - Inspired by traditional engineering methods, where planning and budgeting is key to winning contracts, and where late changes are expensive and unpopular.
- **Not good for game design.**

Sturdy vs Agile

- Differences in practice are much less than differences in rhetoric.
- Example: Boehm's Curve.



- Need a methodology that responds to this issue.

Agile Methodologies

- "Agile" is a relative term.
  - Lots of small steps, with continuous testing and refactoring.
  - Underlying philosophy: Change is inevitable, so plan for it.
    - Or: "No battle plan ever survives contact with the enemy"
      - Helmuth von Moltke
  - Refers to the sturdy camp (unflatteringly) as BDUF ("Big Design Up Front")
  - Inspired by open source and 1990s web development.

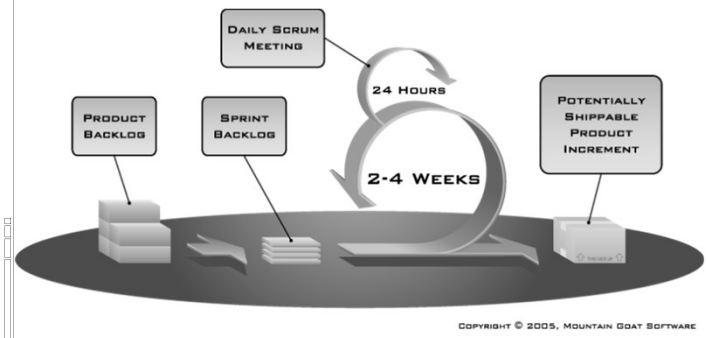
Agile Motivation

- Agile home ground:
  - Low criticality
  - Senior developers
  - Requirements that change often
  - Small number of developers
  - Culture that thrives on chaos
- **Common to game design.**

Implications for you

- So what does this mean for you (individual), you (group) and your project?
  - Your system is *always* buildable and releasable.
    - Easy for us to loop over your commits and check this ☺
  - Regular meetings *with minutes*.
    - Both with client and within group.
  - Issue tracking and prioritization.
  - Comprehensive testing.
  - Refactoring.
  - Collective code ownership.
    - We reserve the right to quiz you on *your team's code* as a part of any exercise.

Scrum at a Glance



Elements of Scrum

- Product Backlog:
  - A list of things the team hasn't yet implemented
  - Stays forever there until project is done
  - Each item has a priority
  - Visible by client
- Sprint Backlog:
  - Basically, a "to do" list
  - Smaller list of things needed to be done
  - Property of the team
  - Due by next sprint

More Elements of Scrum

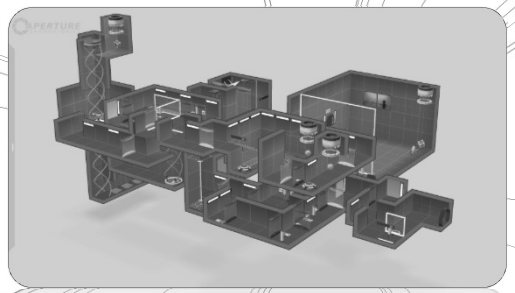
- Sprint:
  - Typically 2-4 weeks long (hence the 30 days)
  - During this time the team gets to finish the sprint backlog items
  - No one is allowed to change the sprint backlog during a sprint
  - Ideally, a team should every 24 hours discuss where they are at in the development

Justification for Scrum

- Scrum takes into account:
  - Things will fall apart because humans are involved.
    - Accept that this problem can't be solved.
    - Work to instead get as much done as possible.
  - Tools
    - Does not discriminate (backlog can be in Excel)
  - Organization
    - Lots of communication is required (lots of meetings/updates)
    - Everybody needs to know how the project is doing.

Today's Activity

Designing a Tutorial Level!



Tutorial Level Reminder

- Tutorial levels are meant to introduce the player to a few main things:
  - How to control the player character,
  - How to interact with the world,
  - What the goals are,
  - What approaches can be used to achieve those goals.

Shown here: Tomb Raider & Fallout 3

Step #1: General Decisions

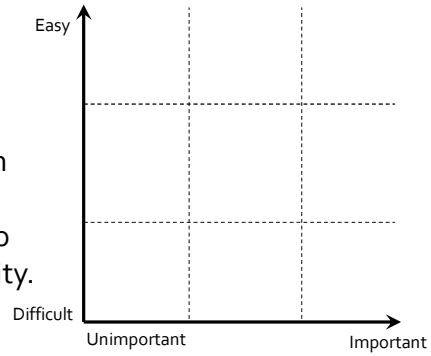
- For each of the following, come up with 3-4 ideas before decising on the next one:
  - What is the general idea of your tutorial level?
  - What will it look like?
  - How will the player be introduced to the different elements of the gameplay?
    - Controls, interacting with the world, goal definition, etc.

Step #2: Feature Lists

- List all the elements that you will need for your tutorial level:
  - Examples:
    - "*Animating main character*"
    - "*Creating level objects*"
- Then, break these down into atomic components:
  - Examples:
    - "*Creating Image #1 for walk animation*"
    - "*Creating coin for tutorial level*"
    - "*Creating interactions for coin objects*"

Feature Lists (cont'd)

- On a piece of paper, create the grid below:
- Number each task and plot it on the grid:
- Then draw contours from top-right to bottom-left to indicate priority.



Step #3: Setting SMART Goals

- SMART goals typically define goals as:
- Specific
  - Well defined
  - Clear to anyone that has a basic knowledge of the project
- Measurable
  - Know if the goal is obtainable and how far away completion is
  - Find out when you have achieved your goal
- Agreed Upon or Attainable
  - Agreement with all the stakeholders what the goals should be
- Realistic
  - Within the availability of resources, knowledge and time
- Time-Based
  - Enough time to achieve the goal
  - Not too much time, which can affect project performance

Setting SMART Goals

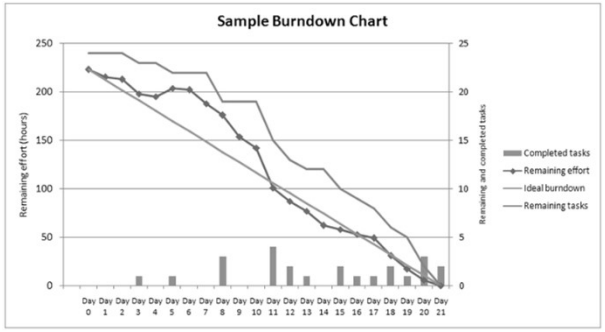
- Create a shared table like the following:

Task	Time Needed (minutes)	Priority (1 = high, 10 = low)	Person Responsible
Finding sprites for walking animation	10	2	Amy
Finding animated coin and star sprites	10	3	Bob
Creating scripts for collecting coins	20	1	Carl
…	…	…	…

Scrum Example: Thanksgiving Dinner

- Roles
- User stories
  - No technical details.
  - When complete, no further consultation with user necessary.
- Product backlog
  - User-side requirements.
  - User sets priority of each item.
- Sprint goals, backlog
  - Developer-side items.
  - Team sets priority of items (prioritization chart).
- Burn down chart
  - Chart of work remaining in sprint backlog.

Burn Down Chart



- If Steve was more diligent, this would be filled with dinner items. Oh well.