Project Management

Project Management

- One of, if not the most important, aspect of software development
- Also one of the most neglected in the hurry to get down to implementation
  - Programmers aren't good at it
- We'll describe something light and easy, suitable for your type of project
  - Inspired by Agile methods

Background

- Large-scale software development & IT projects are plagued with high failure rates:
  - Late
  - Over budget
  - Low quality
  - Product does not meet customer's actual need
- Why? Lots of reasons have been proposed:
  - Writing software is essentially solving problems
  - High degree of uncertainty (requirements, platform, process)
  - Tendency to over-engineer and/or lose focus
  - Integration problems – incompatible platforms, 3rd party etc.
  - Building a plane while flying it

Is games development similar?

- Yes & no
- Similarities:
  - It's software
  - The general phases of development are the same
  - Many of the problems are the same:
    - Bad estimates
    - Changing requirements
    - Poor tracking
- There are also differences…

Is games development different?

- Fun is the primary goal but hard to pin down
  - Gameplay is emergent, unlike storytelling
  - Serendipitous outcome from collaboration between multiple disciplines
  - Can be hard to tell if you're on the right track until everything is in place
  - Inventive nature of the work
  - Frequent changes in direction
    - You never know if the code you write will be thrown out
- Hard deadlines:
  - Launch date often determined by external factors

Code, design and art

- A game is really three projects running in parallel
  - Cross-dependencies can be large & constantly changing
  - Work to minimize them
    - Designers and artists shouldn't need code work to get new content into game
    - Coders don’t need final art to implement a feature
    - Make sure schedule changes in one area don’t hold up another
- Having some things solid before starting anything else is great
  - Design before production start would be great (preproduction, MVP)
  - Studios working on sequels have solid technology before even beginning content creation
  - Can purchase tech to get a bit of a head start

Schools of thought

- Control the uncertainty:
  - Heavy upfront design
  - Sign-offs
  - Build to specification
  - Structured customer involvement
- Embrace the uncertainty:
  - Iterative & adaptive processes
  - People-oriented processes
  - Frequent deployment & feedback
  - Collaborative customer relations

One Agile approach to scheduling

- The creative nature of game development resists heavy up-front, on paper design
  - Iterative methods embrace the chaos more effectively
- We've broken up your assignments into several external milestones
- Goal: a working version of a game that incrementally converges to the final product
- So, how could you go about working on your milestones?

Creating a feature backlog

- Define the features that comprise your game
  - Organize them top-down
  - Derive this from your high concept (assignment #1)
- Choose an appropriate level of abstraction
  - Put down as much detail as is relevant for now
- The whole team should agree with this list
- Recruit feature owners
- Agree on a process for updating this list as things change
  - Who is the keeper?
  - Do we meet to discuss every X weeks?
  - How will we share this list?

Estimation and prioritization

- For **all** the features, estimate time to complete **all** aspects of the feature
  - Coding, tuning, testing, documentation, integration
  - Be conservative, pencil in some time for learning
  - Adjustments can and will be made throughout the project
  - Consult with us if you need help
- Determine how important the feature is to the quality of the final product (impact)
- Priority is a function of cost (time) to complete and the impact to the project
  - Watch for high-impact, low-cost features (high priority),and low-impact, high-cost features (low priority)
- Reflect priority in your backlog (optional)

Risk management process

- Identified risk can be managed:
  - Known knowns, known unknowns, etc
  - Unknown risk will bite you at some point
- Identify risk:
  - Regular discussion and reviews
- Deal with risk **proactively**, i.e. eliminate it up-front:
  - Attack the riskiest tasks first, but time-box them
  - Have a back-up plan
  - Or leave them as “wishlist”
- Deal with it **reactively**, i.e. when it happens
  - Requires clearly defined triggers
  - Requires a contingency plan upfront
  - Needs padding to execute the backup plan
- Add extra padding for risks that you haven't identified

Your project risks

- Name a few?

Setting goals & deliverables

- Internal vs external milestones
- Get together as a team
  - Examine the feature list
  - Agree on what features can be completed for the milestone
    - What needs to be done first
    - How much you should aim for
    - Create a milestone backlog
- Agree on who will drive each feature
  - Take ownership
  - Take initiative
- Record everything – paper or software

Task breakdown & estimates

- Before implementation, break down tasks into chunks (between two hours and three days)
- If a feature requires technical design, make this a task
- If a feature needs special investigation, make this a task e.g. “figure out & prototype how to connect a PC controller”
- Don’t forget non-code content: art, audio
- Don’t forget about integration, testing & tuning
- Try to balance out responsibility & tasks so everyone finishes at the same time

Task breakdown: gotchas

- Beware of vague tasks
  - E.g. “Graphics Engine: 2 weeks”
- If a task estimate is longer than 3 days, break it up into smaller subtasks:
  - “Graphics Engine: 3 weeks” as a **summary** of:
    - mesh rendering: 3 days
    - background rendering: 1 day
    - vehicle rendering: 3 days
    - text system: 2 days
    - etc.

Dependencies and bottlenecks

- Task A has to be completed before work on task B can begin
  - Dependencies lengthen the schedule
  - Exist across tasks and across people
  - Create bottlenecks
  - Need to be anticipated and taken into account when scheduling
  - Project management tools can show you this graphically (in theory)
- Constant vigilance
- Tight communication

Creating a milestone schedule

- First pass:
  - List all tasks to be completed in breakdown
  - Determine who will complete each task
  - Add up completion estimates
  - Agree on order of tasks
  - Set some intermediate due-dates
- This will generate an initial completion date for the milestone at the current level of scope
- The first estimate will be:
  - Very late
  - Wildly optimistic

Scope the feature backlog

- Scoping is the process of dropping tasks/features of the game to make the milestone schedule achievable
- Start with lowest-priority features first and keep cutting until the total estimate fits within the milestone
- If there is nothing left to cut, you go into crunch mode
  - Hopefully that experience will encourage you to scope more drastically next time ;)

Tracking progress – scrum style

- Get together every day for a 15-minute meeting
- Everyone answers three questions:
  - What did you do since the last meeting?
  - What are you doing until the next meeting?
  - Is there anything impeding your progress?
- The first two questions are used to track the progress of the milestone (are you late?)
- The third identifies issues that require additional action (recall risk analysis)
- Leave paper trail
- Stand up ;)

Adjustment

- What do you do when things start to go off the rails?
  - Use up some of the contingency time
  - Steal time from another task or milestone
  - Work longer hours to "make up the time"
  - ► Redefine the task (reduce scope) ◄
- Projects that are behind schedule stay that way unless decisive action is taken to fix the problem
  - "*Adding human resources to a late software project makes it later*" - Fred Brooks
  - Daily meetings catch problems quickly
- Excessive overtime creates stress, degrades morale, and ultimately lowers the quality of the product
  - The only way to avoid it is by being proactive

Shipping a milestone

- Give yourself a couple of days before the milestone date for integration and testing
  - Last minute feature additions will destabilize the build
  - Code & content cut-offs?
- Get ready for the next one
  - Were any features incomplete?
    - Throw them back onto the master feature list
  - Conduct a post-mortem, adjust the process
  - Be honest about your velocities
  - Revisit master feature list
  - Examine priorities, make adjustments, estimate with new refined knowledge
  - If adding features to the game, use a zero-sum approach
    - An equivalent cost feature has to be removed
- Repeat!

Summary

- It takes effort to keep a project on schedule
- You can’t completely control uncertainty and risk
  - Stay flexible
  - Re-examine the schedule often
  - Be prepared to make tough decisions