CS3505/5020
Software Practice II

Quiz return
Project #6 Q/A
Project #7 Overview
Crystal Clear review
Scrum
Quiz return

- **Quiz #1**
  - Average: 17.8
  - Median: 18

- **Quiz #2**
  - Average: 17.2
  - Median: 17

- I will spend a few minutes in class answering questions about the quiz grading.
Project #6 – Completing the Multiplayer Protocol

- Clarifications – Classroom discussion on vague points (Student lead / student resolved)
  - Counting frames
  - Latency
  - Adjusting latency
  - Empty input frames
Project #6 – Completing the Multiplayer Protocol

- This slide is for notes/clarifications made during class:
Project #7

- Write a game (or other pre-approved application) using XNA.
  - Approximately four weeks of work for four team members – should be substantial work to get full credit.

- Teams of 4
  - I may add/allow a 5th member to some teams of 4.

- Projects due during finals week.
  - We will plan on having your last checkpoint during the previous week. You are not expected to do significant work during finals week, although your team may elect to.
Project #7

- Teams must use software processes during this project.
  - Your team will commit to one next Tuesday.
  - This will be a graded component of the project.

- Expect weekly meetings with the TA to record your progress.
  - We will set up a schedule / signup sheet

- Expect ‘daily’ stand-up meetings with your team at the beginning of each class period.
  - Required
Project #7

- TA’s will talk about their experiences in previous years.
Project #7 – Getting started

● Next Tuesday teams must be formed.
  – The entire lecture will be devoted to team meetings, exercises, and software process planning.
  – Teams will be required to submit their project and process plans.
  – Teams will commit to their project at this point.
  – The in-class exercises will count as quiz #3.

● Next Wednesday you will have your first meeting with a TA.
  – TAs will offer recommendations on project scope / duration.
Project #7 – Requirements

● Each game must implement two of the following features for a C grade (three for a B, four for an A):
  – Scrolling view
  – 3-D objects
  – AI that can fairly beat a rookie
  – Multiplayer support
  – XBox compatibility
  – Customizable games / persistent state
  – Storylines that affect game play
  – Animated sprites / sprite sheets

● Additional features may be added to this list.
  – Exact details will be posted on the project web page.
Project #7 – Requirements

- Each team must use scrumworks to plan and record their progress.
  - Teams will put a list of tasks in a backlog.
  - Each week teams should commit to a ‘sprint’ to complete some tasks from the backlog.
  - We will talk about scrum / scrumworks on Thursday.

- All team members must participate.
Crystal Clear

- The Crystal Clear methodology describes:
  - agile processes
  - seven properties of successful teams
    » reviewed today
1. Frequent Delivery

- Why?
- How often?
- What if you can’t deliver to end client?
2. Reflective Improvement

- Why meet and discuss what works and what doesn’t?
- Why post lists so everyone can see what’s working?
3. Osmotic Communication

- What is “Osmotic communication”
- What locality is required?
- What about irrelevant discussions?
- What is a cone of silence?
4. Personal Safety

- Why is it important to speak when something is bothering you?
- What is meant by personal safety?
5. Focus

- What is meant by focus?
- How do you determine what is important?
- Can multitasking be a part of this?
- Should interruptions be allowed?
6. Easy Access to Expert Users

- What is it for?
- How often is it needed?
- How can this be addressed?
7. Technical Environment with Automated Tests, Configuration Management, and Frequent Integration

- Why automate tests?
- What is configuration management?
- Why does frequent integration help?
Crystal Clear Strategies and Techniques

- As an agile method, Crystal Clear does not prescribe specific actions or processes.
  - It does, however, enumerate popular techniques found in XP, Scrum, etc.

- Instead, a list of strategies and techniques are proposed from which a team may select their ideal processes.

- These are an overview – for more details see “Crystal Clear, A Human-Powered Methodology for Small Teams”
**Strategy 1: Exploratory 360°**

- This is applied at the start of a project
- Explore:
  - Business value of the project
  - Requirements
  - Project domain model
  - Technology
  - Project plan
  - Teams
  - Software processes (methodology)
Strategy 2: Early victory

- Mostly a motivational tool

- This is usually the first piece of *visibly* running, functional code

- Do easy things first, hard things second

- Downside?
Strategy 3: Walking skeleton

- An end-to-end implementation of a complete system
  - Missing many elements of functionality

- Different than early victory in that the complete project is visible

- Architecture and functionality evolve together

- Note that this is different than a ‘spike’, which is just a technical proof of concept.
Strategy 4: Incremental Rearchitecture

- This is just a fancy name for continuous refactoring.

- You usually have three choices:
  
  - Predict every architecture requirement in advance
  
  - Incrementally improve system architecture while developing
  
  - Stop development for a system-wide redesign of architecture
Strategy 5: Information Radiators

- The idea is to display information about the project where people will see it.
  - Similar to osmotic communication
  - Bulletin boards with project timelines, tasks, user stories, status information, etc.
  - Updated frequently with valuable information

- These serve two purposes:
  - Managers can quickly see what a team is working on
  - Team members can easily monitor their own progress and may see information relevant to their work
Technique 1: Methodology Shaping

- Have a preliminary meeting / Methodology Shaping Workshop
  - No more than an hour long
  - Start with the given rules for a project and write them down
  - Each team member lists all of the development techniques that they have experienced and disliked and would like to avoid
  - Make another list of experienced things that they liked and would like to repeat
  - Take the liked/repeat list and fit in with the given rules
  - Discuss and combine into team lists
  - Each team member gets 7 votes for each list to say what they think are important (you can spread your votes around or put many on one item – up to seven total)
  - Take the relevant disliked list and put what is relevant on a list of things to avoid
  - Edit and agree on each list to come up with the starting rules for the project
Technique 2: Reflection Workshop

- At the end of each appropriate period have a reflection workshop
- Keep them short (maybe 15 to 30 minutes)
- First input is the methodology shaping workshop list
- Discuss what is working and not working
- Discuss any new ideas that maybe should be tried
- Create new, edited version of the list
Technique 3: Blitz Planning

- Gather attendees (stakeholders, developers, etc.)
- Grab cards and write tasks (or user stories) on them
  - Partition into short tasks (few hours)
  - Do as fast as you can, get done in at most 15 minutes
- Lay out dependencies
  - Sequential tasks – top to bottom
  - Parallel – side to side
- Review tasks (remove duplicates, add as needed)
- Estimate tasks (time and who will do it)
  - If external requirements are needed, write that on
- Sort the tasks
  - Mark sequential dependencies
- Mark the walking skeleton, first release, final release
  - Walking skeleton – tiny implementation that is “end to end”
Technique 4: Delphi Estimation

- Process to estimate the scope, time, people needs, and release plan for a project
- Gather designers and other senior people
- Input use cases for system
- Create table of “factors” for the system in rows
  - Such as: screens, use cases, technical classes, business classes
  - Columns were for each participant
- Iterate filling out table and add up factors to get size estimate
- Determine what people skills are needed and link back to factors
- Determine velocity of people skill per size and compute total time
- Determine releases
- We will not be doing this for our project
Technique 5: Daily Stand-up Meeting

- Short meeting of entire team
- Identify problems (don’t discuss them)
  - If you start discussing, move it to after the meeting with only those needed to be involved staying
- Answer three questions:
  1. What did I work on yesterday?
  2. What will I work on today?
  3. What is getting in my way?

  - Quickly identifies if people are on task or if problems need to be resolved
Technique 6: Essential Interaction Design

- It is easy to overlook usability in favor of functionality
- Think about roles of the users of the system
- Work through these roles and how they will interact with the system
- Factors such as user speed and information availability are important here
- Specify a UI theme or ‘personality’ that will meet the roles
- Define a testing strategy to insure UI competency
Technique 7: Process Miniature

- Basic concept is to run through the process that you have in a very small session
  - From an hour to a day
- Solve a small problem using the same methods as you will for your project
- Gives everyone insight into how you will work to get to where you are going
**Technique 8: Side-by-Side Programming**

- Sit next to each other
  - Close enough to see the screen of the other
  - But work on individual task assignments
  - Can ask at any time to have other look at screen, run a test, or get some help
  - When done, you go back to working on your project

- Effectively does “peer code reviewing”

- Can do some pair programming if you want

- Example
  - Client / server app with one doing client, other doing server
  - Work on it and get to a point: “Hey, I have a stub just finished, can you test it?”
Technique 9: Burn Charts

- Use a burn chart to help you plan and track progress
  - Scrumworks will build these for you

![Burn Chart Diagram]
Scrum

- An agile process that has gained a lot of popularity
  - Not just a software world philosophy
- Scrum parallels many of the techniques from Crystal Clear
  - Thus the Crystal Clear philosophy towards good practice over rigid dogma
- Perfect for small teams
  - 6-9 people + a scrum master
  - Great for disposable apps
  - Not so great where architecture is critical
Scrum

- **Participants**
  - Product owners / stakeholders
  - Scrum master
  - Team members (developers, testers, etc.)

- **Design structure**
  - User stories describe the product functionality
  - Backlog is a collection of undone user stories
  - Sprint is a subset of the user stories in the backlog
  - Release is a collection of completed user stories
Scrum

- **Product backlog**
  - List of features that the system should, some day have
  - Each feature takes roughly 1 to 20 days to build
    » In our case, we are going to use less than 4 hours per feature
  - Product Owner (“Customer”) is responsible for prioritizing list of features

- **Scrum Sprint – 2 to 4 week iteration**
  - Begins with meeting where product backlog is examined and items chosen
  - Scrum team takes each item and breaks into Scrum Backlog (roughly no more than 16 hours of work)
    » Includes estimates of how much work

- **Product backlog items for a particular sprint don’t change during the sprint**
Scrum - 2

- **Daily Scrum**
  - Only pigs (team) can talk
  - Standard standup process
  - IS NOT A STATUS MEETING
    » Designed to foster communication

- **Opportunity to “Inspect and Adapt”**
  - Like cooking when you season, taste, season, ...
  - If things aren’t working, then communication ensues between relevant parties
  - Not the entire team

- **New tasks may be discovered and added to Sprint Backlog (do this Sprint) or tossed up to Product Backlog (do future Sprint)**
Scrum - 3

- **Sprint Backlog** includes estimates of how much work is left to be done
  - Team members are required to keep updating that number at least daily
  - Does not reflect how much work has gone into the task as that is not relevant

- **At the end of a Sprint** have a Sprint Review
  - Team demonstrates software produced
  - Two hour limit on prep time for meeting & no PowerPoint
  - Items that are discovered are added to Product Backlog

- **We will use Scrumworks**
  - [http://hedwig.eng.utah.edu:8080/scrumworks](http://hedwig.eng.utah.edu:8080/scrumworks)
  - I need to create accounts before you log in
What is Our Plan?

- Preparation for next Tuesday
  - Each person create your list of process likes and dislikes and bring to class.
    » Bring something suitable for turning in.
  - Have a meeting and decide what you want to build for the final project.
    » Again, bring something suitable for turning in.

- Tuesday – Short methodology shaping workshop (25 quiz points) followed by Blitz Planning Session (25 homework points)

- Meet with a TA on Wednesday

- Project #7 and agile process officially starts
Examples:

- Stand up meetings
- Test-last
- Osmotic communication
- Email interaction
- SVN
- Minimum meetings
- Individual roles
- Agile development
- Side-by-side programming
- CVS
- Long meetings
- Test-first
- Excessive comments
- Burn-down charts

- Heavy, up-front documentation
- Pair Programming
- Meet physically vs virtually
- Task allocation techniques
- Communication methods
- CRC cards
- Use cases
- Procrastination
- Hack and slash
- Forms of documentation
- White box testing
- Black box testing
- Google Groups
- Mailing lists vs email list
- ...

THE UNIVERSITY OF UTAH