Software Engineering in Game Design

Anne Gatchell - 23 March 2012

What We Will Cover

- What sets game development apart from other forms of software development?
- Briefly introduce game design and the concept of fun in terms of Angry Birds
- Look at a proposed model for the game development process

Game Design: A Melting Pot of Disciplines

- In game design, people from a large variety of backgrounds come together to create a cohesive, compelling product
- People from:
  - Art, music, graphics, human factors, psychology, computer science, and engineering (Callele)
- Issues can arise and individual goals can differ
  - Engineers may be too willing to compromise on the art in order to get the game out on time
  - Artists may not understand the limits of Artificial Intelligence (Callele)

The Elusive(?) Product

- Games are supposed to be fun and entertaining
- Finding the fun is not straightforward, and many attempt to create fun games and fail
- It is possible to get all the technical aspects correct, but the game will not be fun
- Likewise, the graphics can be amazing, but that does not make the game addictive (Norneby, Olsson)

Angry Birds: A Simple Game?

- People love to be envious of Rovio for making a fortune over “simple game” (Mauro)
- Rovio was on the verge of bankruptcy in early 2009 (Cheshire)
- Luckily, Mikael and Niklas Hed recognized the potential for smartphone entertainment
- Set out to methodically create an empire for mobile phones to save the company and create its own intellectual property

Angry Birds: A Simple Game? (continued)

- One March afternoon a game designer there named Jaakko Iisalo showed them a screenshot with a “cartoon flock of round birds, trudging along the ground, moving toward a pile of colorful blocks. They looked cross.”
- “People saw this picture and it was just magical,” said Niklas
- But, Jaakko had pitched hundreds of ideas in the two months prior to that

Angry Birds: A Simple Game? (continued)

- The team worked on the game for eight months, making thousands of changes, and nearly abandoned the project
- But, then Niklas' mother burned the Christmas turkey because she was distracted by the game
- She doesn't play any games
- They realized that they had found the fun

What Can Angry Birds Teach Us?

1) Many things about good game design, which we will not explore in this presentation
2) Finding the fun is not as easy as the product makes it look, much like ballet
3) The team had to repeatedly adjust the game until it was right

How Do We Do Game Development?

- How do Games compare to other areas of software engineering?
- What is the best way to interface between the designers and the programmers?
- How can we meet ever increasing customer expectation?

No Silver Bullet!

- That sounds familiar!
- Just as Fredrick Brooks, Jr. said of software engineering in 1987, there appears to be no game development process to rule them all (Pashley)
- It is good to keep in mind that different processes may work for different people and different team sizes
- Looks like Agile/Extreme styles are beginning to dominate

Well, What Can We Do, Then?

- Johanned Norneby and Tobias Olsson, veteran game engineers from Massive Entertainment:
- Many of the problems in Game Engineering are analogous to regular Software Engineering problems
- And, as has helped with Software Engineering, Game Engineering can be helped by implementing best practices
- (Norneby, Olsson)

Norneby and Olsson Best Practices

1) Iterative Development
2) Manage Requirements
3) Manage Change
4) Verify Quality

(1) Iterative Development

- As we saw in the Angry Birds example, even a simple game will go through a lot of changes on the way to market and on the way to finding the fun
- This lends itself well to Iterative Development, which is a cornerstone of Agile/Extreme programming

(1) Iterative Development (cont.)

- Iterative Development:
  - The game should be developed in short iterations
  - Each iteration will address some aspect of the game to improve
  - At the end of an iteration, there is a working piece of software for delivery to the end user or the focus group

(1) Iterative Development (cont.)

- Why does this lend itself to games?
- Saves time and money:
  - Finding the fun in a game is not as easy as writing down a plan. To get a real answer about how the game will play, we need a working prototype that we can try
  - The sooner we find out that a game is fundamentally not fun, we can kill the project
  - We can get feedback from focus groups right away to make corrections to our game to make it better

(1) Iterative Development (cont.)

- Since the game design is necessarily evolving as we give prototypes to the user and get feedback, we know that the requirements will be changing throughout the entire process
- No need to spend a large portion of time on a requirements document that cannot possibly predict what the user will think and will probably be obsolete within a few iterations

(2) Manage Requirements

- Since our big game design document is now obsolete, we realize that the design of the game is now a process that lasts for the entire production of the game
- Everyone on the team is now responsible for managing requirements
- Instead of a large requirements document, everyone should be familiar with the Vision of the project

(2) Manage Requirements (cont.)

- The Vision should be a very concise and informative description of the essentials of the game
  - What is the game about
  - What the play does most of the time
  - Why is the player doing this
  - What is the surrounding environment
  - What feelings should the game evoke
- Much like a company's vision, the project vision is something that everyone in the team can digest quickly
- If you don't have a concise, clear vision, the game is too undefined and is too risky

(2) Manage Requirements (cont.)

- Other requirements to consider in the development are:
  - Quality Requirements of the Game (fun factor, feelings of player)
  - Business and Organizational Goals of the Company (Rovio: save company, make game that will dominate the iOS store, portable to other platforms after iOS store is dominated, become Disney 2) (Cheshire)
  - User Personas (who is using this game? Rovio: Who uses the iPhone? Everyone. Game must appeal to everyone) (Cheshire)

(3) Manage Change

- Change in software is inevitable
- This is why it is SOFTware and not HARDware
- Therefore, you can't quite treat it like other engineering disciplines
  - Especially in Game Development, which has the fun factor to contend with
- Norneby and Olsson would try to create code that was resilient to change, but the changes the game designers would request always trumped their best efforts

(3) Manage Change (cont.)

- They one day realized that they cannot fight the change. So…
- It is Okay to throw out code
- It is Okay to change an interface that has been around for a long time
- In their attempts to be pro-change with Object-Oriented Programming and flexible code, they were actually being inflexible
- Because, Gamers and Game Designers will always want something out of the ordinary

(3) Manage Change (cont.)

- They found that they should just focus on the functionality at hand
- The problem was that they were trying to predict change and they were considering code reuse
- But, when a programmer thinks about code reuse, they are no longer considering the functionality at hand. They are trying to predict the future
  - As we now know, this is impossible in games

(3) Manage Change (cont.)

- The conclusion about code reuse:
  - "Truly re-usable software elements are discovered, not designed"
- Never design for reuse. It is speculative, and therefore looking outside the scope for the project
- This, happily, made coding fun again at Massive Entertainment
- They saw positive changes in their whole process within a few days, and people were writing better code

(3) Manage Change (cont.)

- Don’t forget to document change, though
- Otherwise, a game could go in circles

(4) Verify Quality

- Find problems as early as possible
- Keep entire system testable as easily as possible
- Treat user testing professionally

This All Makes Sense

- Best practices of Norneby and Olsson seem very logical
- Tried and true methods by seasoned veterans
- But, some argue that more time should be spent in preproduction phase
  - More preproduction should be done
  - More requirements engineering
  - “I would like to see the day that 25 to 40% of… overall prerelease time [is spent in] preproduction” – Game Designer and Producer Eric Bethke (Callele)

More Time in Preproduction?

- Given that it is inarguable that the game design cannot possibly be final before the game is actually complete
- The desire of developers like Bethke appears to be wishful thinking and fear of change, much like Norneby and Olsson had prior to their epiphany
- It would be fantastic if a game could be fully defined before it ever reached a programmer
- That would mean that software engineering is actually quite simple and straightforward, which is not the case

Conclusion: Agile in Game Programming

- Game Development is an enormous topic
- We have not even touched upon game design theories
- Given that games are a particular piece of software in which the fun factor can really only be tested using executable code
- Game programming appears to be best suited to Agile-like environments
- Games are a special area of software engineering, but despite its special nature, it can still apply many of the lessons of software engineering

People Trying Agile

- Many resources to learn about people's game design techniques
- Experiential learning is great!
- Eg. Jeremy Keller describes his small team's experience with trying Scrum and their eventual migration to Lean development and the Kanban board
- http://technitai.wordpress.com/2011/03/14/the-skinny-on-agile-game-design-part-1/

Game Design Tools

- From my searching, there are a multitude of resources available regarding Game Design, which was not the topic of this presentation, but may be of interest
- Game Design Methods: http://gamedesigntools.blogspot.com/p/game-design-methods.html -- A large collection of links all about facets of Game Design

Sources

- Cheshire, Tom. *In depth: How Rovio made Angry Birds a winner (and what's next).* Wired UK. 07 March 11. http://www.wired.co.uk/magazine/archive/2011/04/features/how-rovio-made-angry-birds-a-winner?page=all
- Pashley, Simeon. *Myth of the Silver Bullet Game Production Process*. 3 June 2010. http://game-linchpin.com/2010/06/myth-of-silver-bullet-process.html
- Norneby, Johannes and Olsson, Tobias. *A New Attitude To Game Engineering: Embrace Change, Re-Use, Fun*. 6 August 2009. Gamasutra: The Art & Business of Making Games. http://www.gamasutra.com/view/feature/132491/a_new_attitude_to_game_.php
- Callele, Neufeld, Schneider. *Requirements Engineering and the Creative Process in the Video Game Industry*. 2005 13th IEEE International Conference on Requirements Engineering.

Software Engineering in Game Design

- Game design is a melting pot of many different disciplines which brings together people who might otherwise never work together. Art, music, graphics, computer science, psychology, etc.
- In addition, game design requires the team to create something **fun**, which is a requirement that is more elusive than most ordinary requirements
- With these differences in mind, one may wonder if software engineering practices like Agile have a place in game design.
- Taking lessons from some experts in the field, we explore how iterative development is essential and how many approaches in Extreme/Agile programming are even more valid in Game Programming, because **fun** is such an elusive quality