
CMSC 498M: Chapter 14 Development and Production

Source:

- Lecture notes by Michael J. Katchabaw of U. of Western Ontario.

Overview:

- Early Development: From concept to proposal
- Preproduction: From proposal to prototype
- Production: From prototype to product
- Problems: When development goes wrong

Chapter 14, Slide 1
Copyright © David Mount and Amitabh Varshney

Overview

- Early Development: From concept to proposal
- Preproduction: From proposal to prototype
- Production: From prototype to product
- Problems: When development goes wrong

Chapter 14, Slide 2
Copyright © David Mount and Amitabh Varshney

The First Idea

Idea: Most games begin with a single idea, which can revolve about:

- A character
- A setting
- A story
- A style of gameplay
- A philosophy
- A new technology
- ...and so on

How Original?

- Sometimes completely original, but more often built on **existing work**.
- Totally new ideas are unproven, and consequently untested.
- New variations on something established is **safer**, and more easily accepted by publishers.

Chapter 14, Slide 3
Copyright © David Mount and Amitabh Varshney

Concept Development

Concept development:

- Take initial idea, **refine** it, and **flesh it out**.
- Decide on story and character elements, gameplay, setting, etc.
- Develop concept art.
- Prepare a **pitch document**, followed by a **detailed proposal**.
- You will not likely be funded ... few people get paid to sit and think!

Elements: Think about publisher's needs:

- Publishing strategy (many inexpensive games or a few more expensive ones).
- Risk tolerance.
- Scheduling constraints.
- Licensing conditions and issues.
- Preferred platform and technologies.
- Type of game wanted (totally new, sequel, conversion, adaptation).

Chapter 14, Slide 4
Copyright © David Mount and Amitabh Varshney

The Concept Document (or Pitch Document)

Concept Document:

- Convey the **goals** and purpose of the game.
- Helps management (or prospective publisher) assess **viability**.
- The purpose is to **sell the game** concept to key decisions makers.
- Should be brief, approximately **five pages** in length, more or less.

Topics:

Premise: (or High Concept) What the game is about.

Player Motivation: What will drive the player on?

Unique Selling Proposition (USP): What will make it stand out?

Target Market: What age/gender group? General or niche?

Genre: (and any twists on standard genres)

Target Rating: ESRB rating (Everyone, Teen, Mature, ...)

Target Platform: (and resources required)

License: Use of licensed characters (e.g., James Bond)

Chapter 14, Slide 5
Copyright © David Mount and Amitabh Varshney

The Project Proposal

Project Proposal:

- Follow-up to the concept document, providing **further details**.
- Longer than the concept document, **ten to twenty pages**.

Includes: in addition to Concept-Document topics:

Hooks: Most attractive features (visuals, gameplay, story, ...)

Gameplay Mechanics: List 10-20 elements describing the experience.

Online Features: Multiplayer? Community? Scale and finder services?

Technology: Software or hardware technologies. Built or purchased?

Art and Audio Features: E.g., licensed music, motion capture?

Story and Characters: Outline of plot and main characters.

Production Details: Development team, budget, and schedule.

Risk Analysis: Anticipated risk areas and potential costs.

Concept Art: Major characters, scenes, user interface.

Chapter 14, Slide 6
Copyright © David Mount and Amitabh Varshney

Overview

- Early Development: From concept to proposal
- **Preproduction: From proposal to prototype**
- Production: From prototype to product
- Problems: When development goes wrong

Chapter 14, Slide 7
Copyright © David Mount and Amitabh Varshney

Preproduction

Preproduction:

- Getting ready for the development of the game.
- Complete the game design, produce suitable documentation, and do technical prototyping to demonstrate its feasibility.

You need to provide a proof of concept:

- Preproduction basically **proves** your team can make the game and that the game is **worth making**.
- If you cannot do this successfully, you and your idea may be written off in favor of something else.

Chapter 14, Slide 8
Copyright © David Mount and Amitabh Varshney

Preproduction Documentation

Preproduction Documentation:

- Several documents are written during the preproduction phase.
- To flesh out and **formalize initial proposal** ideas and concepts.
- Provide a **blueprint** for when the game goes into development.

These include:

- Game design document
- Art bible
- Production path
- Technical design document
- The project plan

Chapter 14, Slide 9
Copyright © David Mount and Amitabh Varshney

The Game Design Document

Game Design Document:

- By the end of preproduction, you should have a game design document **detailing everything** that will happen in your game.
- Equivalent to **functional specs** in traditional software development.
- **Living Document**: Will change frequently and evolve over time.

Elements:

- Overview
- Story
- Gameplay mechanics
- Game-world behavior
- Game elements
- Game progression
- System menus

Game design document

- Overview
- Story
- Gameplay mechanics
- Game-world behavior
- Game elements
- Game progression
- System menus

Art bible
Production path
Technical design
The project plan

Chapter 14, Slide 10
Copyright © David Mount and Amitabh Varshney

The Game Design Document: The Overview

The Overview:

- A **single-page summary** of the game's design.
- Will help newcomers become familiar with the game.

Should include:

- The game's high concept or focus.
- A one paragraph summary of the story.
- Key gameplay features and other important gameplay aspects.
- Summary of game's innovations and reasons for success.

Game design document
• Overview
• Story
• Gameplay mechanics
• Game-world behavior
• Game elements
• Game progression
• System menus
Art bible
Production path
Technical design
The project plan

Chapter 14, Slide 11
Copyright © David Mount and Amitabh Varshney

The Game Design Document: The Story

The Story:

- Easy-to-read narrative of what transpires in the game.

Includes:

- Game setting.
- Key plot elements (e.g., divided into a three-act structure).
- Back story (history).
- The main characters.
- Non-player characters, including villains, those supporting the player, and those that are neutral.

Game design document
• Overview
• Story
• Gameplay mechanics
• Game-world behavior
• Game elements
• Game progression
• System menus
Art bible
Production path
Technical design
The project plan

Chapter 14, Slide 12
Copyright © David Mount and Amitabh Varshney

The Game Design Document: Gameplay Mechanics

Game mechanics:

- Describes how the player will **interact** with the game world.
- What **actions** the player can carry out.
- What the **results** of these actions are.

Information to include:

Genre statement: Including any new twists the game makes, and how the game uses or departs from genre conventions.

Player capabilities: Be as specific. Describe everything the player can do in the game and how the player does it.

User interface: Interaction modes and so on.

Initial start-up activities: For creating/customizing the players' characters.

Maintenance activities: That the player does with their characters throughout the game.
...and anything else that seems important.

Game design document

- Overview
- Story
- **Gameplay mechanics**
- Game-world behavior
- Game elements
- Game progression
- System menus

Art bible
Production path
Technical design
The project plan

Chapter 14, Slide 13
Copyright © David Mount and Amitabh Varshney

The Game Design Document: Game-World Behavior

Game World Behavior:

- This section documents how the **game world reacts** to the players' actions.
- Complements the game mechanics section.

Includes:

- How will **NPCs react to the player**? What will they do in which situations? How are they triggered?
- How will **NPCs act when the player is not around**? How do **NPCs interact with one another**?
- How do other elements in the game world react to the player?

Tips:

- Be as specific as possible. The more questions you answer, the more likely you get what you want.

Game design document

- Overview
- Story
- Gameplay mechanics
- **Game-world behavior**
- Game elements
- Game progression
- System menus

Art bible
Production path
Technical design
The project plan

Chapter 14, Slide 14
Copyright © David Mount and Amitabh Varshney

The Game Design Document: Game Elements

Game Elements:

- Includes characters, **items used or wielded** by the player and non-player characters, and other objects and mechanisms.

Main Element Types:

- **Characters:** Active, non-player controlled elements. E.g., villains.
- **Items:** Things that the player can pick up and use or manipulate in some fashion. E.g., weapons.
- **Objects/Mechanisms:** Things that operate, but not be picked up by player. E.g., doors and puzzles.

Be sure to include:

- Physical descriptions.
- Behavioral or operational descriptions.
- Definitions of relationships to other elements.
- Comparisons to other elements.
- Concept art, if available.

Game design document
• Overview
• Story
• Gameplay mechanics
• Game-world behavior
• **Game elements**
• Game progression
• System menus
Art bible
Production path
Technical design
The project plan

Chapter 14, Slide 15
Copyright © David Mount and Amitabh Varshney

The Game Design Document: Game Progression

Game Progression:

- Breaks the game down into the **events the player experiences**, and how they change and progress over time.
- Very strong correlation with how the story unfolds.
- In many games, this is broken down on a **per-level basis**.

Includes: For for each level or stage of the game:

- Structure and organization.
- How it will look, sound, and feel to the player.
- The major challenges, obstacles, or puzzles faced by the player.
- The part of the story contained within it.
- Player's Experience: Difficulty, experiences, and emotions felt.

Game design document
• Overview
• Story
• Gameplay mechanics
• Game-world behavior
• Game elements
• **Game progression**
• System menus
Art bible
Production path
Technical design
The project plan

Chapter 14, Slide 16
Copyright © David Mount and Amitabh Varshney

The Game Design Document: System Menus

System Menus:

- This is where you describe the **menus, options screens**, and other screens presented to the player outside the game itself.
- Since these do not have a direct impact on gameplay, they should be discussed in their own section.

Includes:

- Functionality and features available in the menus and screens.
- How these menus and screens flow into each other in the game.
- How the user interfaces with these options.

Game design document
• Overview
• Story
• Gameplay mechanics
• Game-world behavior
• Game elements
• Game progression
• **System menus**
Art bible
Production path
Technical design
The project plan

Chapter 14, Slide 17
Copyright © David Mount and Amitabh Varshney

The Art Bible

The Art Bible:

- During preproduction, it is important to establish a **consistent look** and style for the game as early as possible.
- Can be pencil sketches, but colored glossies have a bigger impact.
- Notes and annotations of the artwork should also be included for additional references.
- Include descriptions of **artistic styles**, directions, instructions, and limitations.
- Can also be the source for **story boards** and other concept art included in the design document.

Game design document
• Overview
• Story
• Gameplay mechanics
• Game-world behavior
• Game elements
• Game progression
• System menus
Art bible
Production path
Technical design
The project plan

Chapter 14, Slide 18
Copyright © David Mount and Amitabh Varshney

The Production Path

Production Path:

- Explains how to go from **concept to implementation**.

Includes:

- Art tools
- Modelers and rendering tools
- Level editors and design tools
- Music and sound tools
- Game engines
- Software development tools, ...

Note:

- All of these tools must be **compatible!**
- This must be worked out now so that costs and timings in acquiring the tools can be factored into the project plan.

Game design document
• Overview
• Story
• Gameplay mechanics
• Game-world behavior
• Game elements
• Game progression
• System menus
Art bible
Production path
Technical design
The project plan

Chapter 14, Slide 19
Copyright © David Mount and Amitabh Varshney

The Technical Design Document

Technical Design Document:

- Complements the game design document discussed earlier.
Game design document: Describes how the game will function.
Technical design document: Describes how that functionality will be implemented.

Includes:

- Software design and code structure.
- Artificial intelligence
- Animation and graphics
- Sound
- Networking
- ... and other technologies used in implementing the game.

Game design document
• Overview
• Story
• Gameplay mechanics
• Game-world behavior
• Game elements
• Game progression
• System menus
Art bible
Production path
Technical design
The project plan

Chapter 14, Slide 20
Copyright © David Mount and Amitabh Varshney

The Project Plan

Project Plan:

- Roadmap describing how the game is going to be built.
- Tasks to be completed.
- Dependencies among these tasks.
- Overhead hours.
- Use all of this to develop a schedule.

Includes:

- Resource plan
- Budget
- Schedule and milestones to track progress.

Tips:

- Software project planning tools may help.
- Must be revised and updated throughout.

Game design document

- Overview
- Story
- Gameplay mechanics
- Game-world behavior
- Game elements
- Game progression
- System menus

Art bible
Production path
Technical design
[The project plan](#)

Chapter 14, Slide 21
Copyright © David Mount and Amitabh Varshney

The Project Plan: The Constraint Triangle

Constraint Triangle:

Ideal: Development is free, built instantly, have perfect quality.

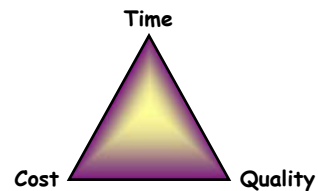
Reality: Time, cost, and quality must be traded off.

Tradeoffs:

Decrease time: Adding more personnel (costing more money) or by reducing quality.

Reduce costs: Fewer developers (and increasing development time) or by reducing quality.

Increase quality: Need more developers or more time to do so.



Chapter 14, Slide 22
Copyright © David Mount and Amitabh Varshney

The Prototype

Prototype:

- The **tangible end result of preproduction** is the game prototype.
- A working piece of software that captures the **game's essence**.
- What makes it special, better than the rest, and what will turn it into a hit.
- It is important to capture the **look and feel** of the game properly.
- Shows the vision of the game.
- Proves that you are **effective** in going from ideas to reality.

Tips:

- Pulling off a good prototype is hard. Assets and technology do not yet exist.
- Cheaply simulate aspects of the game.

Chapter 14, Slide 23
Copyright © David Mount and Amitabh Varshney

Overview

- Early Development: From concept to proposal
- Preproduction: From proposal to prototype
- **Production: From prototype to product**
- Problems: When development goes wrong

Chapter 14, Slide 24
Copyright © David Mount and Amitabh Varshney

Production

Production:

- After preproduction results in a prototype or technology demonstrations, and these are accepted, you are free to proceed with production.

Major Elements:

Development: of the game based on the results from preproduction.

Testing: of the game.

Release: to manufacture.

Maintenance: after release.

Chapter 14, Slide 25
Copyright © David Mount and Amitabh Varshney

Development

Development:

- The long haul of video game production.
- For modern video games typically lasts **six months to two years**.
- Very little can be done well in less than six months; there is simply too much to do.
- If longer than two years, you risk your game going stale or becoming obsolete before it is released.

Time Management:

- Initially it seems you have more than enough time.
- As deadlines approach, reality sets in, followed by **panic**.
- Imperative to break large tasks into **small manageable tasks** that can be rigorously tracked.
- **Track progress** constantly.

Chapter 14, Slide 26
Copyright © David Mount and Amitabh Varshney

Development

Agile Software Development: Growing interest in games industry.

- Rapid, **continuous delivery** of useful software.
- Working software is **delivered frequently** (weeks versus months).
- Working software is the principal measure of progress.
- Even late changes in requirements are welcomed.
- Face-to-face conversation is the best form of communication.
- Projects built around motivated individuals, who should be trusted.
- Continuous attention to **technical excellence** and **good design**.
- Simplicity.
- **Self-organizing teams**.
- Regular **adaptation** to changing circumstances.

Chapter 14, Slide 27
Copyright © David Mount and Amitabh Varshney

Development

Survival tips:

- Maintain good **communication** across the development team.
- Keep design documentation **up to date**.
- Maintain the **team's identity and spirit**.
- Give marketing and public relations the **materials** and **demos** they need - they will help keep people's spirits up when things get tough.
- Be ready for a **shock** or two. When these happen, keep your head down and do the work! Things are rarely as bad as they seem.
- Have a **few features ready to throw away** to help manage scope in the long run.

Chapter 14, Slide 28
Copyright © David Mount and Amitabh Varshney

Testing

Testing:

- Important for both **validation and verification** purposes.
- Should occur throughout development to remove problems as soon as possible.

Verification:

- Are we building the **game right**?
- To eliminate bugs, remove imbalances, and so on.

Validation:

- Are we building the **right game**?
- To improve game design, gameplay, and so on.

Chapter 14, Slide 29
Copyright © David Mount and Amitabh Varshney

Testing: Different Types of Testing

Unit testing:

- The testing of game **modules** on an individual basis.

System testing:

- The testing of **integrated game modules** as a more-or-less complete system.

Acceptance testing:

- An essentially **complete game** is demonstrated for acceptance and publishing.

Chapter 14, Slide 30
Copyright © David Mount and Amitabh Varshney

Testing: Different Types of Testing

Alpha:

- Internal testing.
- The game is **mostly playable** from start to finish.
- Basics are complete. Some content and gameplay **might be missing**.
- The focus shifts from building to **finishing**; from creating to **polishing**.

Beta:

- Internal or external.
- Everything is now complete. An essentially **finalized game**.
- Goal is to stabilize and eliminate remaining bugs before release.
- If possible, **public beta** gets a lot of extra testing for little cost.
- The last portion of beta testing is **crunch time**, where the only important thing is finishing the game.

Chapter 14, Slide 31
Copyright © David Mount and Amitabh Varshney

Testing: Different Types of Testing

Black-box (functional) testing:

- Game functionality is tested according to **specification**, without looking at its internals.

White-box (structural) testing:

- The game is tested according to its **internal structure** and code to ensure it behaves correctly when provided with test data.

Regression testing:

- Developing libraries of test cases that the game is sent through each time a **change or update** is made.
- The purpose here is to retest the game to ensure it still works correctly **after modifications**.
- Can be applied to both black-box and white-box testing equally well.

Chapter 14, Slide 32
Copyright © David Mount and Amitabh Varshney

Release and Maintenance

Release:

- The game is released to manufacture when one of the candidate releases has been **thoroughly tested** and deemed acceptable.
- For console games, may require **approval of console manufacturer**.
- Sometimes, this is referred to as "going gold".
- You can finally **celebrate**.

Maintenance:

- **Patches:**
 - Fix **bugs** discovered after release.
 - Handle **incompatibilities** with user hardware/software configurations.
- **Upgrades and updates:**
 - Additional content to **enhance** the original game. Can be new levels, characters, weapons, story elements, and so on.
 - These are really **mini-projects**, and need to be handled as such.

Chapter 14, Slide 33
Copyright © David Mount and Amitabh Varshney

Overview

- Early Development: From concept to proposal
- Preproduction: From proposal to prototype
- Production: From prototype to product
- **Problems: When development goes wrong**

Chapter 14, Slide 34
Copyright © David Mount and Amitabh Varshney

When Development Goes Wrong

Sometimes Good Development goes Bad:

- Even if you plan well and follow good development models and processes, things can still go wrong.
- Certain problems routinely arise over the course of development.
 - If you know about them in **advance**, you can often **prevent them**.
 - When you cannot prevent them, you can at least **prepare in advance** to minimize or control their effects.

Chapter 14, Slide 35
Copyright © David Mount and Amitabh Varshney

Classic Mistakes

Overly optimistic schedules:

- Goals will constantly be missed, and you will lose the confidence and respect of team members, as well as your publisher.

Feature creep:

- If too many features are allowed into the project midstream, milestones will be missed and the game could bloat out of control.

Undermined motivation:

- Game developers are not usually motivated by the same things that inspire the general workforce. Find ways to motivate your team.

Weak personnel:

- If you pick the wrong people, your game project might be doomed from the start.

Uncontrolled problem personnel:

- Deal with disruptive team members before they have a negative impact on the project.

Chapter 14, Slide 36
Copyright © David Mount and Amitabh Varshney

Classic Mistakes

Poor work environment:

- If you work in an environment that is noisy, poorly lit, and so on, work may be adversely affected as a result.

Contractor failure:

- If you are relying on an external group for technology or content, and they do not pull through, you can be in trouble.

Requirements gold plating:

- If your project is too ambitious in too many areas, you are likely headed for trouble.

Insufficient management control:

- You must be able to track your progress in meaningful ways, or you could be in big trouble and not even realize it.

Waiting too long to fix bugs:

- Fix bugs quickly. Lingered bugs can cause other problems.

Chapter 14, Slide 37
Copyright © David Mount and Amitabh Varshney

Classic Mistakes

Omitting necessary tasks from estimates:

- Make sure to include everything in your schedule, even those things that seem unimportant (meetings, interviewing new hires, code reviews, creating screen shots, and so on).

Misunderstood tasks:

- Avoid confusion. Make sure that all tasks are well understood by all.

Distributed development teams:

- Geographically dispersed teams must work extra hard on communication.

The "Not Invented Here" syndrome:

- Sometimes better to buy outside software than build it yourselves.

Pop-up tasks:

- Plan for unexpected glitches and changes in requirements.

Chapter 14, Slide 38
Copyright © David Mount and Amitabh Varshney

Ineffective Recovery Strategies

Ineffective Recovery Strategies:

- When projects start to slip, managers typically resort to one of these strategies:
 - Plan to **catch up later**.
 - Require **mandatory overtime**.
 - **Add people** to the project.
 - Hold more **meetings**.
 - Close your eyes and **make a wish**.
- Naturally, these strategies are not very effective!

Chapter 14, Slide 39
Copyright © David Mount and Amitabh Varshney

Effective Recovery Strategies

Scale Back:

- The most effective way to reduce a project's schedule is to **decrease its scope**.
- If people know ahead of time a feature might get cut, they will be less emotional if it has to be killed.
- **Plan carefully**: Don't leave holes in your design by killing a feature.

Prioritize:

- Finish **high priority tasks** first, even if they are not the flashiest.
- Team members will be motivated to work efficiently so they get to the tasks on their wish list.
- Create the **most important content first**. You are less likely to waste money creating content that never gets used.

Chapter 14, Slide 40
Copyright © David Mount and Amitabh Varshney

Effective Recovery Strategies

Use a combination of strategies to be more effective.

- Keep things manageable.
- Keep everyone motivated and interested in the project.
- Eliminate problem personnel.
- Create a developer-friendly atmosphere.
- Identify core features and eliminate activities that are not geared towards delivering them.
- Most importantly, avoid the classic mistakes!

Chapter 14, Slide 41
Copyright © David Mount and Amitabh Varshney

Summary

Summary:

- Early Development: From concept to proposal
- Preproduction: From proposal to prototype
- Production: From prototype to product
- Problems: When development goes wrong

Chapter 14, Slide 42
Copyright © David Mount and Amitabh Varshney