Al-Nahrain University

College of Sciences

Computer Science Department

Al-Nahrain University - College of Science 1988

Computer Game Design and Programming

Dr. Safaa H. Shwail

Lecture 1 - Introduction

What is a Game?

A game is a type of play activity, conducted in the context of a pretended reality, in which the participant(s) try to achieve at least one arbitrary, nontrivial goal by acting in accordance with rules.

What Is Computer Game Design?

Computer Game design is a large field, drawing from the fields of computer science/programming, creative writing, and graphic design. Game designers take the creative lead in imagining and bringing to life video game worlds.

A Brief History of Computer Games

To design the computer games of the future, you need to understand the games of the present. And the games of the present are rooted in the games of the past. To this end we will give a brief overview of the most important developments in computer games over the past fifty years.

The changes over the past fifty years can globally be categorized in a number of different types, which all influenced each other:

- **Changes in the hardware for playing games.** These developments had a huge effect on what was possible in computer games and, hence, enabled the game designers to make different and more fascinating games. We went from devices with minimal computing power, memory, and graphics to the consoles of today with special 3D graphics cards, DVD disks to store game data, and Internet connectivity.
- **Changes in interaction devices.** Over the years the devices we use to control the games have also changed. Early game consoles had special rotating knobs or simple joysticks and a few buttons. Nowadays game controllers have multiple joysticks, and lots of buttons. And in recent years this has been extended with devices that measure movement, like the Wii controllers of Nintendo or the Kinect system of Microsoft. Obviously these had a strong impact on the game play.
- **Changes in the software tools available.** Initially game developers wrote every line of code themselves (often in assembly language) and drew every pixel of the artwork. Nowadays extensive game engines and other middleware packages are available allowing for much more sophisticated games. And artist, animators, and level designers use advanced tools that help them to create complicated artwork efficiently.
- **Changes in the game business.** Game companies have considerably changed over the past fifty years. While initially games were developed

primarily by individuals, nowadays there are huge teams of specialists working on a single game. Development budgets have grown from a few thousand dollars to tens of millions of dollars per game. Also educational programs have appeared to train the professionals needed by the game industry.

- **Changed in the demographics of the players.** While initially games were primarily played by young males, this has considerably changed in recent years. The number of female players is close to the number of male players, and the age of players ranges from 4 to 100. This has led to new genres of games.
- **Diversification.** Initially, computer games were primarily played on arcade machines. Over the years the variety of platforms increased considerably. Nowadays we play games on consoles, personal computers, handheld devices, telephones, TV set-top boxes, etc. Also the type of players has diversified, as has the ways people like to play (for example casual, online, or hard-core). All of these devices and ways of play come with their own business models and development budgets.
- **Changes in the design of games.** All these changes have in turn led to considerable changes in the design of games. Game designers used the new hardware and interaction devices to create new forms of immersive gameplay. They created games that attracted different demographics. And they started to better understand better what makes an interesting game.

Chronology

Arising as a prominent branch of game development in the 1970s after the huge success of arcade video games, game designers as we know them today were tasked with designing the bulk of content for the game, including the rules, storyline, characters and overall appeal. Today, game designing is a multi-million dollar industry that’s only expected to grow larger as technology advances. Take a look at the timeline below to see how the industry has evolved and expanded over the years.

- **1952** - Willy Higinbotham creates what is commonly referred to as “the first video game.” Similar to table tennis, this 2 person game was played on an oscilloscope.

- **1961** - A MIT student, Steve Russell, creates the first interactive game, Spacewars, played on mainframe computers.

- **1971** - Computer Space, developed by Nolan Bushnell and Ted Dabney (founders of Atari), becomes the first video arcade game released.

Although it was instantly popular, many people found the game too difficult.

- **1972** – Realizing the potential of video games, Magnavox released Odyssey, the first home video gaming system. Most notably though, Atari is founded and quickly recognized as the leader in the video game industry. Their first released game, Pong, was wildly successful and soon became available as a home version.
- **1974** – Steve Jobs, one of Atari's technicians and later a circuit board creator, presented an idea to the Atari founders for a personal computer system. Because funds were tied up in other projects, Bushnell referred a venture capitalist to Jobs for funding support. That personal computer, of course, was the beginning of Apple.
- **1975** – The first computer game hits the markets. Gunfight used a microprocessor instead of hardwired circuits.
- **1977** – Retailing at $249.95, a large chunk of money at the time, the Atari 2600 game console is released.
- **1978** – Adding another level of competition and appeal to video games, Space Invaders hits arcades as the first game to track and display high scores. Soon after, the game Asteroids took it a step further and allowed three letter initials to be stored with top scores.
- **1980** – The first 3D game, Battlezone, is created. The game caught the eye of the US Government, who later modified it for training exercises. Due to the advancing complexity of games, companies begin to form teams to specifically address game design. Game designers and game programmers soon became separate, distinct careers.
- **1981** – The gaming industry proves its prominence with the first dedicated periodical, Electronic Games.
- **1985** – Developed by a Russian programmer, Tetris is released for arcades, video game consoles, as well as home computers.
- **1989** – Game Boys, handheld gaming devices made by Nintendo, hit the market to much avail. Later in the year, Sega releases the Genesis game console.
- **1994** – The Entertainment Software Rating Board is created due to concerns about violence in games and the marketing tactics used. Games now receive a rating displayed on the packaging.

- **1995** – Sony releases the PlayStation in the U.S. By 1997, 20 million units had been sold.
- **1996** – Arcades switch their focus from traditional video games to the more popular physical riding games, such as skiing, and car/bike racing.
- **1998** – The Sega Dreamcast is released, one of Sega's last pulls to stay in the market.
- **2000** – Sony's PlayStation 2 is released. The 500,000 initial units sell out instantly at $300 apiece. The same year, The Sims surpasses Myst as the best selling PC game. Game designers now work in teams of dozens to quickly create the complex games being sold.
- **2001** – "Next Generation" gaming systems are introduced. The Microsoft Xbox and Nintendo GameCube are not only more interactive for the user, but also easier to develop games for. Shortly after the release of these products, Sega announces it will no longer produce hardware.
- **2004** – The Nintendo DS is released as a purposefully portable system. Sony follows a year later with their portable version, the Sony PSP.
- **2006** – The Nintendo Wii revolutionizes the market with its controller system, designed to mimic actual physical movements such as swinging a tennis racket or throwing a bowling ball. During the same year, the PlayStation 3 is released as the most sophisticated (and expensive) console.
- **2007** – Apple releases the iPhone, creating an entirely new device in which to play games.
- **2008** – The App Store is introduced. With a diverse array of functionalities, games quickly become the most popular and lucrative "apps." Game designing and developing for Smartphone applications becomes a large niche. In a successful effort to get people of all ages involved and excited about the Wii, Nintendo releases the Wii Fit game. By the next year, Wii Sports surpasses Super Mario Bros as the bestselling video game with over 40 million units sold.
- **2011** – Projected to grow an additional 30% by 2016, the gaming industry produces sales of over 18 billion per year. Colleges and degrees specifically for game design and production are becoming increasingly advanced and popular.

Top Game Production Companies

Who actually makes all those games you know so well? Who leads the $18 billion video game industry? Although large companies take up the bulk of market space, independent producers and at-home coders are definitely making their presence known. Below, we take a look at some of the top game production companies:

Activision Blizzard

Call of Duty, Crash Bandicoot, Guitar Hero, James Bond, Shrek, Spider-Man, Spyro, StarCraft, Tony Hawk, Transformers, World of Warcraft, X-Men.

EA Sports

Battlefield, Burnout, FIFA, Harry Potter series, Madden NFL series, Medal of Honor, NASCAR series, NCAA Football series, NHL, Rock Band, SimCity, The Sims series.

Nintendo

Batman, Blades of Steel, Duck Hunt, Final Fantasy, Iron Sword: Wizards & Warriors, Mario, Mega Man, Super Mario Bros., Teenage Mutant Ninja Turtles, The Legend of Zelda.

Microsoft

Brute Force, FreeCell, Game Chest series, Golf, Halo, Joy Ride, Microsoft Flight Simulator, Monster Truck Madness, Tetris, Tut's Tomb, Zoo Tycoon.

Sony

A Bug's Life, Atlantis: Lost Empire, ESPN Extreme Games, Formula One 2000, Grand Theft Auto: Vice City, Grand Turismo, Treasure Planet, Twisted Metal, Wipeout.

Sega

Alien Storm, Formula One World Championship: Beyond the Limit, Galaxy Force, Michael Jackson's Moonwalker, Pro Wrestling, Sega Tetris, Sonic the Hedgehog, Time Traveler, World Grand Prix, Zombie Revenge

Warner Brothers Interactive

300: March to Glory, Demolition Man, Mortal Kombat, Speed Racer, Terminator Salvation, The Lord of the Rings: Aragom's Quest.

Lecture 2- What players want and expect-designer skills

Why do players play?

The first question we should consider is: why do players play games? Why do they choose to turn on their computer or console and run spider instead of visiting the art museum or going to see a movie? What is unique about computer games versus other human entertainment media? What do games offer that other activities do not? It is by understanding what is attractive about games that other media do not offer. Then we can try to emphasize the differences that separate our art form from others. To be successful, our games need to take these differences and play them up, exploiting them to make the best gameplay experience possible.

A. What players want?

1. Players Want a Challenge

Many players enjoy playing games because they provide a challenge. This provides one of the primary motivating factors for single-player home games, where social or bragging rights motivations are less of an issue. Games can entertain players over time, differently each time they play, while engaging their minds in an entirely different way than a book, movie, or other form of art.

2. Players Want to Socialize

For most people, the primary reason they play games is to have a social experience with their friends or family. People like to play these games because they enjoy spending time with their friends and want to engage in a shared activity that is more social than going to a movie or watching TV.

Multi-player games are basically adaptations of single-player games into multi-player incarnations. These games typically provide a single-player game in addition to a multi-player game, both played with nearly the same set of rules and game mechanics. But even in these single-player-turned-multi-player games, players like to socialize while playing.

A separate category of multi-player games is what has come to be called Massively Multi-Player (MMP) games. These games tend to be more in the style of role-playing games, where players wander around “virtual worlds” and meet and interact with the other characters in these worlds, characters that are controlled by other players. These games tend to be played over large networks such as the Internet, instead of over LANs, and as a result players only socialize with each other through what they type into the computer.

3. Players Want a Dynamic Solitary Experience

Some game players are looking for a social experience, while others are looking for something dynamic that they can engage in by themselves. Sometimes friends are not available to play, or players are tired of their friends, or simply are tired of having to talk to other people all the time. Similar to the difference between goings to a movie theater with an audience versus renting a video alone at home, the antisocial nature of single-player games attracts a lot of people who have had enough of the other members of the human race.

4. Players Want Bragging Rights

Particularly in multi-player gaming, players play games to win respect. Even in single-player games, players will talk with their friends about how they finished one game or about how good they are at another. Players will brag about how they played the whole game through on the hardest difficulty in only a few hours.

Looking at games both old and new, the high-score table and the ability to enter one’s name into the game, even if only three letters, provides a tremendous incentive for people to play a game repeatedly. Players, who may not have much to brag about in their ordinary lives, can go down to the arcade and point out to all their friends their initials in a game.

5. Players Want an Emotional Experience

As with other forms of entertainment, players may be seeking some form of emotional payoff when they play a computer game. The emotions that games are able to evoke in players are much stronger than what can be experienced in other media where the experience is less immersive and considerably less personally involving.

6. Players Want to Explore

One of the main motivating forces that propels players through many level-based games is the desire to explore new spaces and see new environments. Anyone who has played a progression-based game knows the feeling of getting to a new and different level and wanting to just look around for a few moments before taking on the objectives at hand.

7. Players Want to Fantasize

A major component of the popularity of storytelling art forms is the element of fantasy. Many people, sometimes wants to get away from their own normal lives and escape to an altogether different world. So, computer games have the potential to be an even more immersive form of escapism.

8. Players Want to Interact

Games have found their greatest successes when they have played up the interactive nature of the experience and provided our audience with something they cannot get anywhere else. Game designers need to constantly keep this in mind as they are developing their games if they are to have any chance of winning players' attention.

B. What Do Players Expect?

1. Players Expect a Consistent World

As players play a game, they come to understand what actions they are allowed to perform in the world, and what results those actions will produce. Worse still is when the consequences of the players' actions are so unpredictable that players cannot establish any sort of expectation.

2. Players Expect to Understand the Game-World's Bounds

When playing a game, players want to understand which actions are possible and which are not. They do not need to immediately see which actions are needed for a given situation, but they should understand which actions are possible to perform and which are outside the scope of the game's play-space.

3. Players Expect Reasonable Solutions to Work

Once players have spent some time playing a game, they come to understand the bounds of the game-world. They have solved numerous puzzles, and they have seen what sorts of solutions will pay off. Later in the game, then, when faced with a new puzzle, players will see what they regard as a perfectly reasonable solution. If they then try that solution and it fails to work for no good reason, they will be frustrated, and they will feel cheated by the game.

4. Players Expect Direction

Good games are about letting the players do what they want, up to a point. Players want to create their own success stories, their own methods for defeating the game, something that is uniquely theirs. But at the same time, players need to have some idea of what they are supposed to accomplish in this game. Thus, players want to have some idea of what their goal is and be given some suggestion of how they might achieve that goal.

5. Players Expect to Accomplish a Task Incrementally

Once players understand what their goal in the game-world is, they like to know that they are on the right track toward accomplishing that goal. The best way to do this is to provide numerous sub-goals along the way, which are communicated to players just as clearly as the main goal. Players are rewarded for achieving these sub-goals just as they are for the main goal, but with a proportionally smaller reward.

6. Players Expect to Be Immersed

Once players get into a game, they are progressing through various challenges, they have a good understanding of the game's controls, and they are role-playing a fantasy. They have forgotten that they are playing a game at all, just as a film audience may forget they're in a theater or a book's reader may become completely swept up in the lives of the story's characters.

7. Players Expect Some Setbacks

Players tend not to enjoy games that can be played all the way through the first time. If the game is so unchallenging that players can storm right through it on their first attempt, it might as well not be a game. If they wanted something that simple they might as well have watched a movie.

8. Players Expect a Fair Chance

Players may be able to figure out the proper way to overcome the obstacle through trial and error, but there should be some way to figure out a successful path on their first try.

9. Players Expect to Not Need to Repeat Themselves

Once players have accomplished a goal in a game, they do not want to have to repeat their accomplishment. If the designer has created an extremely challenging puzzle, one that is still difficult to complete even after players have solved it once, it should not be overused.

Some games will even automatically save players' games at a newly achieved position, a process sometimes known as checkpoint saving. This method is somewhat superior since often players, having succeeded at an arduous task, will be granted access to a new and exciting area of gameplay, one that they will immediately want to explore and interact with.

10. Players Expect to Not Get Hopelessly Stuck

There should be no time while playing a game that players are incapable of somehow winning, regardless of how unlikely it may actually be. At some games, if players failed to do a particular action at a specific time, or failed to retrieve a small item from a location early in the game, they would be unable to complete the game.

11. Players Expect to Do, Not to Watch

cut-scenes; they can be very useful tools for communicating a game’s story, or for passing along to players information they will need in order to succeed at the next section of gameplay. That said, I do believe that cut-scenes should be stripped down and minimized to the absolute shortest length that is necessary to give some idea of the game’s narrative, if any, and set up the next sequence of gameplay.

What Skills Does a Game Designer Need?

In short, all of them. Almost anything that you can be good at can become a useful skill for a game designer. Here are some of the big ones, listed alphabetically:

1. **Animation** — Modern games are full of characters that need to seem alive. The very word “animation” means “to give life.” Understanding the powers and limits of character animation will let you open the door for clever game design ideas the world has yet to see.
2. **Anthropology** — you will be studying your audience in their natural habitat, trying to figure out their heart’s desire, so that your games might satisfy that desire.
3. **Architecture** — you will be designing more than buildings — you’ll be designing whole cities and worlds. Familiarity with the world of architecture, that is, understanding the relationship between people and spaces, will give you a tremendous leg up in creating game worlds.
4. **Brainstorming** — you will need to create new ideas by the dozens, nay, by the hundreds.
5. **Business** — the game industry is just that, an industry. Most games are made to make money. The better you understand the business end of things, the better chance you have of making the game of your dreams.
6. **Cinematography** — many games will have movies in them. Almost all modern videogames have a virtual camera. You need to understand the art of cinematography if you want to deliver an emotionally compelling experience.
7. **Communication** — you will need to talk with people in every discipline listed here, and even more. You will need to resolve disputes, solve problems of miscommunication, and learn the truth about how your teammates, your client, and your audience really feel about your game.
8. **Creative Writing** — you will be creating entire fictional worlds, populations to live in them, and deciding the events that will happen there.

9. **Economics** — many modern games feature complex economies of game resources. An understanding of the rules of economics can be surprisingly helpful.
10. **Engineering** — Modern videogames involve some of the most complex engineering in the world today, with some titles counting their lines of code in the millions. New technical innovations make new kinds of gameplay possible. Innovative game designers must understand both the limits and the powers that each technology brings.
11. **History** — many games are placed in historical settings. Even ones placed in fantasy settings can draw incredible inspiration from history.
12. **Management** — any time a team works together toward a goal, there must be some management. Good designers can succeed even when management is bad, secretly "managing from below" to get the job done.
13. **Mathematics** — Games are full of mathematics, probability, risk analyses, complex scoring systems, not to mention the mathematics that stands behind computer graphics and computer science in general. A skilled designer must not be afraid to delve into math from time to time.
14. **Music** — Music is the language of the soul. If your games are going to truly touch people, to immerse, and embrace them, they cannot do it without music.
15. **Psychology** — your goal is to make a human being happy. You must understand the workings of the human mind or you are designing in the dark.
16. **Public Speaking** — you will frequently need to present your ideas to a group. Sometimes you will speak to solicit their feedback, sometimes you will speak to persuade them of the genius of your new idea. Whatever the reason, you must be confident, clear, natural, and interesting, or people will be suspicious that you don't know what you are doing.
17. **Sound Design** — Sound is what truly convinces the mind that it is in a place; in other words, "hearing is believing."
18. **Technical Writing** — you need to create documents that clearly describe your complex designs without leaving any holes or gaps.
19. **Visual Arts** — your games will be full of graphic elements. You must be fluent in the language of graphic design and know how to use it to create the feeling you want your game to have.

How could anyone possibly master all of these things? The truth is that no one can. But the more of these things you are comfortable working with, the better off you will be. This is another reason that game designers must be confident and fearless. But there is one skill that is the key to all the others.

20. **Listening**: The most important skill for a game designer is listening. Game designers must listen to many things. These can be grouped into five major categories:
    - **Team:** since you will be building your game and making crucial game design decisions together with them. Remember that big list of skills? Together, your team might have all of them. If you can listen deeply to your team, and truly communicate with them, you will all function as one unit, as if you all shared the same skills.
    - **Audience**: because these are the people who will be playing your game. Ultimately, if they aren’t happy with your game, you have failed. And the only way to know what will make them happy is to listen to them deeply, getting to know them better than they know themselves.
    - **Game:** It means you will get to know your game inside and out. You will get to know what is wrong with your game by listening to it run.
    - **Client:** The client is the one who is paying you to design the game, and if you don’t give them what they want, they’ll go to someone else who does.
    - **Self:** This sounds easy, but for many, it is the most difficult kind of listening. If you can master it, however, it will be one of your most powerful tools, and the secret behind your tremendous creativity.

Lecture 3- Concept Development and Genres

Concept Development and the Game Proposal Document

Before you can begin design work on a game, you must have a general idea of what the game is about, which genre it fits into, and what your publisher's goals are. When you know this, you can create a pitch document with the goal of getting a green light to develop the project.

Concept Development

Most games begin with a single idea. The idea can revolve around a character, gameplay, philosophy, or new technology. The idea might come from a friend, coworker, or publisher, or from deep in your own subconscious. Sometimes the idea is completely original, but more often it builds on the work of game designers who have gone before. Beginning designers often fret about this, but it's not something to worry about. Why? Because usually when game publishers say that they're looking for a "new" idea, what they really want is a new wrinkle on something that's established already.

So the goal of the first phase of development is to come up with your idea, which will eventually evolve into the *high concept* of the game. The high concept is the one-or two-sentence response to the question, "What is your game about?"

Here are some sample high concepts:

- A busty female archaeologist pursues ancient treasure. (*Tomb Raider*)
- Ping-Pong on the computer. (*Pong*)
- An ordinary technician battles trans-dimensional monsters after an accident at a secret research facility. (*Half-Life*)
- A street-racing game where you drive a getaway car for the mob. (*Driver*)

Genres

Your high concept might fall neatly into an existing genre, but more and more games are *hybrids*, combining elements from different categories. If you're creating a game that crosses genres, make sure that you're familiar with the conventions of each genre so that you end up with the best of both, instead of the worst.

The following sections describe various genres. Your game should probably belong to a genre that you enjoy playing yourself, so that you've already internalized the genre's conventions.

1. Adventure Games

Adventures are story-based games that usually rely on puzzle-solving to move the action along. They can be text-based or. They can be told from a first-person perspective, second-person (most text games in which the hero is “you”), or third-person.

Generally, adventure games aren’t in real time, unless they’re an action-adventure hybrid. The player usually takes as much time as he wants between turns, and nothing happens in the game world until he enters a command.

The original adventures were *parser-based*, accepting simple sentence commands from the keyboard. More modern adventures are *point-and-click*, in which the player indicates what he wants to do by moving the mouse around the screen.

Players generally expect an adventure game to have a large, complex world to explore, along with interesting characters and a good story.

2. Action Games

Action games are real-time games in which the player must react quickly to what’s happening on the screen. The category is dominated by first person shooters (FPS) such as *Quake*, *Unreal*, and *Halo*.

The action-adventure hybrid, however, is often a third-person game, such as *Tomb Raider*, in which you can see the hero or heroine as he or she moves through the environment. Typically, the gamer has much more to do than just shoot and kill enemies.

3. Role-Playing Games (RPG)

In role-playing games, the gamer generally directs a group of heroes on a series of quests. Gameplay revolves around gradually increasing the abilities and strengths of these heroes. Classic RPGs include *Ultima*, *Might and Magic*, and *Final Fantasy*.

Like an adventure game, an RPG features a huge world with a gradually unfolding story. Players expect to be able to micromanage their characters, all the way down to the weapons they carry and the specific armor for each part of their bodies. Combat is an important element, by which the heroes gain strength, experience, and money to buy new equipment.

Fantasy RPGs also feature complex magical systems, as well as diverse races of characters that make up the player's party.

4. Strategy Games

Strategy games require players to manage a limited set of resources to achieve a predetermined goal. This resource management frequently involves deciding which kinds of units to create and when to put them into action.

In the classic *Command & Conquer*, for example, the player has to continually balance which kind of unit to build, how much tiberium to harvest, how many resources to allocate to offense and to defense, and so on.

Older strategy games were typically *turn-based*. The player could take his time as he made each decision, and the computer acted only when the player indicated he was ready. Now, real-time strategy (RTS) games set the computer AI in motion against the gamer whether he's ready or not.

Multiplayer versions of RTS games substitute human opponents for the computer's AIs. These games are enormously popular on the Internet.

5. Simulations

Simulations, or sims, are games that seek to emulate the real-world operating conditions of complicated machinery, such as jet fighters, helicopters, tanks, and so on. The more serious the sim, the higher the premium that's placed on absolute accuracy, especially with equipment controls. Players expect to spend hours learning the intricacies of the machine, and they expect a thick manual to help them with the finer points.

6. Sports Games

Sports games let players vicariously participate in their favorite sport, either as a player or a coach. Prowess in a real world sport isn't required for success in its computer-game counterpart, but then, that's sort of the point. One of the things we want from a computer game is wish fulfillment, the opportunity to do things we can't in real life.

7. Fighting Games

Fighting games are two-person games in which each player controls a figure on the screen, using a combination of moves to attack his opponent and defend against his opponent's attacks. These games are generally viewed from a side perspective, and each session lasts only a few minutes.

8. Casual Games

Casual games include adaptations of traditional games such as chess, bridge, hearts, and solitaire. They also include easy-to-play, short-session games on the Web, such as *Slingo, Poker,* and *Concentration*.

Television game shows are also represented in this category, with the very popular *Jeopardy*, *Wheel of Fortune*, and *Who Wants to Be a Millionaire?*

9. God Games

God games (sometimes called *software toys*) are games that have no real goal, other than to encourage the player to fool around with them just to see what happens. Examples include *The Sims* and *RollerCoaster Tycoon*.

Designers in this genre try to create games in which the player can do no wrong. The games are very open-ended, with no correct way to play and no preset winning conditions.

10.Educational Games

Educational games are those that teach while they entertain. Sometimes called *edutainment*, examples of these games include *Oregon Trail* and *Reader Rabbit*.

Generally, these games are aimed at a much younger audience than most commercial products. Their designers work closely with experts on the subject matter to ensure that the content is appropriate for the target group.

11.Puzzle Games

Puzzle games exist purely for the intellectual challenge of problem solving, such as *The Castle of Dr. Brain* and *The Incredible Machine*.

The puzzles are an end in themselves and aren't integrated into a story, as is common in adventure games.

12.Online Games

Online games can include any of the preceding genres, but their distinguishing feature is that they're played over the Internet. Entire communities grow around the most successful of these games, and the designers of games like *Everquest* and *Ultima Online* are constantly creating features that encourage those communities to flourish.

13.Platformer

A platformer involves moving a character in a 2D or 3D environment to overcome impeding obstacles with physical movement, such as running and jumping over a cliff to a safe area on the other side.

The Game Proposal Document

The result of the concept development phase is the game proposal. Think of the proposal as the executive summary of what you want to accomplish. You must quickly convey how the game is played, what will make it great, how long it will take to develop, and how much it will cost. At this stage, the information is imprecise. Later, if the project is approved, you'll refine this information into a project plan. The format of this document changes from company to company, but all formats seem to include the following elements.

- **High Concept**

It's "the vision thing." State what your game is about, in one or two sentences.

- **Genre**

A single sentence that places the game within a genre, or a hybrid of genres.

- **Gameplay**

This section summarizes what the player will do when he's playing the game. Typically, this section leads off by placing the game within a genre, and then explains how it departs from genre conventions in creative and entertaining ways.

- **Features**

List the major selling points of your game. Identify the technical advancements and what they mean to the game (that is, why they'll help sell more units). These selling points are likely to end up as a bulleted list on the back of the game box. It's a shorthand way of telling people what's inside the box.

- **Setting**

Summarize in a few paragraphs what makes your game world and its occupants interesting and unique.

- **Story**

If your game has a story, this is the place for a quick synopsis of what happens.

- **Target Market**

For whom are you developing the game? Is it a niche market of specialized genre fans? The mass market? Kids? Sports fans?

This section should also include some historical information about how games of this type have traditionally sold to the target demographic.

Target Hardware Platforms

Identify the target platforms on which your game will be played. As development costs rise, more and more games are *cross-platform* projects released on several gaming systems to leverage the costs and create a greater chance of financial success. If it’s a PC game, list the target hardware requirements, including memory and processor speed.

Estimated Schedule and Budget

This is one of the hardest parts of the game proposal to put together. Estimating the schedule and budget of a software project is part science, part art. Entire books have been devoted to the topic, and we *still* get it wrong.

Competitive Analysis

What games have come out in this genre? How did they sell? Why will yours do better or worse? What competition will your game have to face when it comes out? How will your game stack up against the competition and survive the fierce battle for shelf space?

The Team

List the major credentials of your team. If your team has been together for a while, this is easy—you just list the games you’ve produced together. If you’re assembling a new team, list the credits for the major team players (designer, tech lead, art lead, and so on).

Document Summary

Reiterate why this will be a great game, and why your team is the one to pull it off. Show that you understand the publisher’s goals for the product, and give him the confidence he needs to say yes.

Lecture 4- Project Lifecycle

Preproduction (Proof of Concept)

The work products of this phase are the game design document, the art production plan, the technical design document, and the project plan (which itself is actually a suite of documents). Preproduction culminates in the delivery of the game prototype, a working piece of software that shows off how fun the game is to play.

1. The Game Design Document

By the end of preproduction, you should have a game design document that exhaustively details everything that will happen in the game. The features in this document then become the requirements from which the art production plan and the technical plan are made.

During the development cycle, the game design document should always be the most current representation of everything there is to know about what the player experiences in the game. This should include complete information about the gameplay, user interface, story, characters, monsters, AI, and everything else, down to the finest detail.

Such a document, if committed to paper, would be the size of a telephone directory, impossible to maintain, read by no one, and almost instantly out of date. Instead, put it on your internal network as a set of Web pages. Maintaining your documents as Wiki pages keeps the design up-to-date, and also gives everyone on the team easy access to everything at all times.

2. The Art Production Plan

Preproduction is when you establish the “look” of your game and decide how the art will be created.

- ***The Art Bible***

During preproduction, the designer, art director, and concept artist collaborate on setting the artistic style of the game. The concept artist makes reference sheets for other artists to work from, and together the team arrives at a unified look. Establishing this art bible early on helps orient new artists coming onto the project, and ensures the final product will have a consistent style throughout. Most of this art can take the form of pencil sketches, but it is often useful in selling the game to develop a few glossy pieces that capture the high concept and pack a good visual punch.

In the early stages of the game, it is also a good idea to assemble a visual reference library of images that reflect the direction you want the art to take. These images can come from anywhere—magazines, travel books, movie posters, and so on—as long as they're used only for guidance and don't find their way into the final product.

- ***Production Path***

The production path is the process by which you go from concept to reality, from an idea in someone's head to actual figures and gameplay on the screen. For example, to create a functioning character in an action game, you must find the most efficient way to go from a designer's spec, to a concept sketch, to a 3D model, to a skin for the model, to animation for the figure, to applying AI to the character, to dropping him into the game and seeing how he works. All the tools you select along the way must be compatible. They must be able to "talk" with each other so that the work you do on one step can be imported to the next step, manipulated, and passed up the line.

- ***Assets, Budgets, Task, and Schedules***

The Production Plan also includes the first draft of the asset list, team tasking, equipment budget and costs, etc. Like the Game Design Document, this plan must be updated and kept current throughout the life of the project.

3. The Technical Design Document

The technical design document sets out the way your tech lead plans to transform the game design from words on a page to software on a machine. It establishes the technical side of the art production path, lays out the tasks of everyone involved in development, and estimates the time to completion of those tasks. (From these man-month estimates, you learn how many people you need on the project and how long they will be with you. This, in turn, has a direct effect on your budget.)

The tech document describes the core tools that will be used to build the game, and it details which of these are already in-house and which have to be bought or created. The document also lists the hardware and software that must be purchased, as well as any changes you need to make to your organization's infrastructure (storage capacity, backup capabilities, network speed) to support development.

4. The Project Plan

The project plan is the roadmap that tells you how you're going to build the game. It starts with the raw task lists in the tech plan, establishes dependencies, adds overhead hours, and turns all that into a real-world schedule. The final project plan is broken down into several independently maintained documents.

- ***Manpower Plan***

The manpower plan is a spreadsheet that lists all the personnel on the project, when they will start, and how much of their salaries will be applied to the project.

- ***Resource Plan***

The resource plan calculates all the external costs of the project. Building from the tech plan, it takes the timing of the hardware purchases to support internal personnel, and it estimates when the external costs (voice, music, video, and so on) will be incurred.

- ***Project Tracking Doc***

This is where you keep track of whether you're on schedule. Some producers use project management software for this, but many find the programs too inflexible to manage all aspects of the game's development.

- ***Budget***

After applying the overhead multipliers to the manpower plan, you combine these numbers with the resource plan to derive your month-by-month cash requirements and the overall budget for the game.

- ***P&L***

The original profit and loss estimate was made during the concept phase. As development progresses and costs become clearer, the P&L must be kept current.

- ***Development Schedule***

Many developers chafe against creating a firm schedule and committing to a specific release date, but you owe it to yourself and your company to do exactly that.

After a release date has been set, a whole different machine goes into motion. The marketing team books advertisements that will appear in the months running up to the release date. The PR (Public Relations) department negotiates with magazines for cover stories, well-timed previews, and feature articles. The sales group commits to end caps in the software stores. Changing the release date of the software is likely to torpedo all the carefully planned efforts of these groups, and your game may sell far fewer units than it could have.

- ***Milestone Definitions***

Milestones are significant points in the development process marked by the completion of a certain amount of work, called a *deliverable*. These deliverables should be concrete and very precisely defined, with language such as “Concept sketches for 15 characters, front, side, and back” or “Weapon #1 modeled, skinned, and operational within the game, with a placeholder sound effect, but without animations or visual effects.” Avoid fuzzy deliverables, such as “Design 25% complete.” The best deliverables are binary: They’re either complete or they’re not, with no ground for argument in-between.

5. **Game Prototype**

The tangible result of preproduction is the prototype. This is a working piece of software that captures onscreen the essence of what sets your game apart from the crowd, and what will turn it into a hit.

This “look and feel” demo can be the single greatest influence on whether the project goes forward. Publishers like to be able to look at a screen and “get it” right away. If they can’t see the vision within a minute or two, they’re unlikely to fund the rest of the project.

The finished prototype not only shows the vision, but also establishes that your production path is working and that you can go from ideas to reality in a reasonable and effective way.

Development

Your development schedule is likely to last six months to two years. Very little software can be designed, coded, and tested in less than six months (although with the advent of Internet gaming and mobile gaming, “small” games are coming back into style). Likewise, games that spend longer than two years in development run the risk of going stale, suffering personnel turnover, having features trumped or stolen by other games, or seeing the technology lapped by hardware advances. Any of these problems can lead to redesign and reworking, which in turn lead to schedule delays.

The trick to surviving this long stretch is to break large tasks into small, manageable tasks that are rigorously tracked. You can’t know whether you’re behind on a project if you don’t track the tasks. You should do this as often as once a week.

One effective task-management technique is to have each developer track his own task list, complete with time estimates. These individual lists roll up into a master list, which shows at a glance the estimated time to completion for the entire project. This method is particularly useful for seeing whether one person's task bar sticks out beyond the others. If this happens, that person is the *de facto* critical path for the project, and you should take a close look at his list to see whether some tasks can be handed to someone else.

If you are an external developer working for a publisher, your progress is tracked for you in the form of contractual milestones. The incentive to stay on schedule is clear—if you don't meet the milestone, you don't get paid.

Alpha

The definition of *alpha* varies from company to company, but generally, it is the point at which the game is mostly playable from start to finish. There might still be a few workarounds or gaps, and not all the assets might be final, but the engine, user interface, and all other major subsystems are complete.

As you enter alpha, the focus starts to shift from building to finishing, from creating to polishing. Now is the time to take a hard look at the features in the game and decide whether any of them must be dropped in order to make the schedule. Now is when more testers come onboard to start ferreting out bugs. Now is the first time the game is seen and evaluated by people outside the development team.

Beta

At beta, all assets are integrated, all development stops, and the only thing that happens thereafter is bug fixing. Stray bits of artwork can be upgraded, bits of text can be rewritten, but the goal at this point is to stabilize the project and eliminate as many bugs as possible before shipping.

If you're creating a console game that's subject to the approval of the console manufacturer, the final weeks of beta will include submissions to that company so their testers can verify that the game meets their own quality standards.

If it is a PC game, it can be sent to an outside testing firm for compatibility testing. This will uncover, you hope, any pieces of hardware (or combinations of hardware) under which the game won't run.

Code Freeze

In the last few days of beta, you are likely to be in a code freeze, when all the hard work is done and the preparation of candidate master discs begins. Each of these discs is sent to testing, and the only changes allowed to the code base are those that specifically address showstopper bugs that turn up in testing.

RTM (Release to Manufacture)

The game is released to manufacture when one of the candidate releases has been thoroughly tested and found to be acceptable. You can finally celebrate.

Patches

On the PC side of the house, it has become almost inevitable that a game will need to be patched after its release. In a world where literally thousands of hardware combinations exist, it is impossible to test all of them. If a consumer finds that his particular combination of BIOS, graphics card, sound card, monitor, CPU, operating system, keyboard, mouse, and joystick causes problems in the game, most developers will work with him to figure out the source of the problem. If the problem is pervasive enough, the developer will issue a patch.

Upgrades

An upgrade is different from a patch. It represents additional content that’s been created to enhance the original game. Companies create upgrades for a number of reasons. In some cases, it is simply to extend the life of the original game. In other cases, it is an effective strategy to keep part of the team gainfully employed while a smaller group goes on to the early stages of their next project.

In any case, an upgrade is a mini-project and needs to be handled like one, with testing, milestones, and all the other elements of good software management.

Lecture 5 - Genre-Specific Game Design Issues I

Action Games

Your goal in an action game is to keep the player moving and involved at all times. You want to create an adrenaline rush that makes his heart pound and his palms sweat. The primary skills the player needs are hand/eye coordination and quick reflexes. Deep thinking is generally not required, although the better games in this category do call for quick tactical thinking on the fly.

- **Point of View**

Your first design choice is to select the point of view. First-person games put the camera in the character's head. The player sees what his character sees. In third person, the camera is outside the main character, usually floating just above and behind but sometimes moving to different positions to provide a better view of the action.

First-person games tend to be faster-paced and more immersive. There's a greater sense of being "in the world," because the player sees and hears along with his character.

Third-person games allow the player to see his character in action. They're less immersive but help the player build a stronger sense of identification with the character he's playing.

First-person games tend to have more beautiful game environments and higher-detail nonplayer characters (NPCs). This is because the game engine doesn't have to devote any of its resources to drawing the main character.

Third-person games chew up a lot of resources drawing the main character and the animations that go along with it, leaving correspondingly fewer resources to render the game world and the creatures in it.

If your main character is a generic "everyman" whose look and identity are not of great importance, consider using first person. However, if your emphasis is on the main character instead of the world around him, consider using third person.

- **Weapons**

Weapons are extremely important in an action game. They must be appropriate to the game fiction you have created, whether fantasy, science fiction, or real world. They must have interesting characteristics that encourage players to master their quirks. They must be well balanced, which means that there cannot be one weapon that will automatically guarantee victory.

Give your player more powerful weapons as the game progresses and opponents become more formidable. These weapons should not only look interesting on the screen but also be accompanied by flashy graphics and sound effects when the player uses them. Although this sounds simple, it requires close coordination between you and the weapons AI programmer, the special effects programmer, the texture artist who creates the weapon, the artist who creates the special effects, and the sound engineer.

Engine

In all probability, you will be working with a licensed 3D engine from a developer such as Epic (*Unreal*), id (*Quake*), or Monolith (*Lithtech*). For console games, you'll be working with middleware supplied by a company such as Criterion (*Renderware*) or Alias (*Check Six*).

Each game engine has its advantages and disadvantages, and you should consider several criteria when making your choice:

- **Ease of use.** Each game engine has its own editor, and each editor has its quirks. Check out the user community for a vigorous discussion of plusses and minuses.
- **Cross-platform capability.** If your game is to appear on more than one piece of hardware (for example, PC and PlayStation 2), your engine must support those platforms.
- **Look-and-feel.** Games developed in a particular engine tend to resemble one another. During the evaluation process, play games that use the various technologies, and see which appeal to you the most.
- **Support.** One engine can have an active user community and receive ongoing support from its creators, whereas another can have poor documentation and be incomprehensible to anyone but the people who wrote it. Again, the user community is helpful here.
- **Availability.** Do you select an engine that is still in development, or one that has already come to market?
- **Extendibility.** How easy is it for your programming team to modify and add features to the engine?
- **Cost.** Engines are expensive, but they're almost always cheaper than building the technology yourself.

RPGs

Role-playing games (RPGs) revolve around characters, story, and combat. They take place in large, expansive worlds and are frequently played out over hundreds of hours.

- **Character Growth**

RPGs have a slow, delicious build that starts the player’s character as a weakling in a strange and dangerous world. Through carefully managed encounters and alliances, the hero and his party slowly grow in competence and power until they are able to take on the baddest of the bad guys.

You must give players a range of choices in the attributes of the characters in their party. Some attributes should be shared by many classes of characters, but a few should be unique to a particular class. This allows players to balance the team yet bias it towards their own style of gameplay.

Although you should allow the player to select his party individually and to allocate attributes among them (within preset limits), you should also be prepared to generate reasonable parties automatically for the player who wants to skip this step.

- **Story**

Generally, storytelling in RPGs is accomplished through a series of quests. As the player carries out the missions, he explores the world and learns more about its inhabitants and his place among them.

To deal with the never-ending conflict between linearity and nonlinearity, don’t give the player the quests one at a time. Instead, group the missions in a series of small clusters so that, although he has a choice of what he is working on at the current moment, he isn’t overwhelmed with too many possibilities. At any given time, the player should have several immediate goals, one or two midterm goals, and one final goal.

- **Combat**

Early RPGs were called *hack-and-slash* games, and combat still plays an important role in this genre. If you don’t handle combat well, you won’t have a good RPG. Here are some quick pointers:

- Design an interface that gracefully handles the encounters and makes the player feel that he’s always in control.

- Whether your game is in real time or is turn-based, give the player a chance to make meaningful choices as the combat progresses.
- Don’t overwhelm the player right away. Give him a training area where he can acquire competence. Carefully sequence his first several combats so that he can win and acquire a sense of mastery.

Adventure Games

The original adventure games combined exploration with puzzle-solving. They were stories in which the player was the hero.

Adventure games have evolved from their static, text-based origins to include more and more real-time elements. Whether you’re designing a “pure” adventure game or an action/adventure hybrid, the defining characteristics of the genre are still story and puzzle.

- Story

If you don’t have a good story, you don’t have a good adventure game. It’s up to you, the game designer, to decide what the story is. You must decide what your story is about and build the player’s activities around that central theme. You must create interesting people, in interesting places, doing interesting things.

- Puzzles

After you have the story, it’s time to create a threat to the hero’s world and put obstacles between him and his goal. These obstacles are the puzzles, and they must flow naturally from the setting and story.

A good puzzle provides a pleasant, temporary frustration that leads directly to the rush of the “AHA!” moment. A bad puzzle leaves the player angry, frustrated, and distrustful of the designer.

Here are some quick tips:

- Each puzzle must be appropriate to its setting, it must be reasonable for the obstacle to be there, and when the player solves it, he should know why what he did worked.
- To get puzzle ideas, think about the villain. He is the one who does not want your hero to succeed—how is he likely to try to interfere with the hero’s progress?
- The puzzles must make sense. Give the player enough clues that he can solve them, don’t make them too hard, and make sure that every puzzle advances the story.

- Don’t think of puzzles as roadblocks to slow down the player so that he “gets more hours of gameplay.” Instead, help him along.
- Every puzzle is a storytelling opportunity.

- **Interface**

Simplify your interface. The interface you design determines the kinds of activities, puzzles, and interactions the player will have—not the other way around.

Your interface must allow the player to do as much as possible with the minimum amount of effort. Effort, in this case, is defined by the number of clicks it takes to perform actions such as talking with people, examining the environment, doing object-on-object interactions, using inventory, and so on.

If allowing a particular interaction means increasing the number of clicks the player must make for all other activities, abandon that interaction.

If there are well-established protocols for the style of game you are building, use them.

The interface is not the place to experiment. The highest compliment a player can pay to the interface is not to notice that it is there.

- **Linearity versus Nonlinearity**

The biggest complaint you hear about adventure games is that they are too linear. Players don’t like it if they can’t make choices. But if you give the player *too* many choices at once, he gets lost.

The solution is to design a linear series of open environments. In each open area, the player has many activities he can pursue in any order he likes. When he has accomplished them all, the designer closes off the area, does a little storytelling, and then moves on to the next.

- **Exploration**

People are curious. They want to see what’s around the next bend. Your adventure game should scratch this itch and take the player on a journey through a landscape of visually interesting places.

Be careful how you dole these out. If you open the entire environment to the player within the first ten minutes of the game, he has no surprises left.

Instead, use access to new places as a reward for the player and to show him that he is making progress. Tease him with hints and glimpses of areas he can't get to. Try to make the first view of each new area memorable.

Strategy Games

The key to strategy game design is balance. Balance is everything, and it can be achieved only through thousands of hours of playing. Play a little, tweak a little; play some more, tweak some more. More than any other genre, it is vital to have a strategy game up and running early so that months can be spent on polish and on play balancing.

This balance extends to everything in the game. There can't be only one right way to do things. No one strategy can always succeed; otherwise, the game quickly ceases to be fun.

In a well-balanced game, one's success is almost completely determined by one's skill.

- **Resources**

Balance the amount of raw material available. Too much, and the player will never need to make decisions about it. Too little, and the player will spend his time worrying about that, to the exclusion of all else.

You must also balance the rate of production. It's no fun if the Red Team can build 100 tanks in five minutes, but the Blue Team can build only three anti-tank bazookas in the same time span.

- **Teams**

Most strategy games have two teams opposing each other, but some have more. Regardless, each side in the game must have an equal chance to win. The good strategy player must feel the same about the teams available at the start of the game. The sides must be balanced.

- **Units and Weapons**

If all your testers wind up using the same weapon or unit, it is too strong, and you need to make it less effective. If there is another weapon that no one uses, it's too weak—beef it up.

Give each unit or weapon a single distinguishing characteristic, both visually (so that the player can quickly pick it out on the screen) and functionally (so that he can categorize its use in his head). In addition to its main function, however, include a few other minor capabilities as well.

You don’t have to give the same units to both sides, but you do have to create a defensive weapon or strategy for every offensive weapon you create.

- **Realism versus Fun**

Do you make it real, or do you make it fun? The answer is almost always to make it fun.

Even though you can base your weapons on actual ordinance, it’s better to make them fun than to have them correspond exactly to their real-world counterparts. The physics of a certain weapon may confine it to a restricted range in the real world, but if it would balance the game better to give it a longer range, do so.

- **Artificial Intelligence (AI)**

If the computer opponent always does the same thing, or is too difficult or too easy, the game will suffer. AI is a job for an expert in the field, not a game designer. Nevertheless, you can help the AI programmer by giving him a clear idea of how you would like the computer opponents to behave in various situations. Talk with him throughout development, and rework designs that he says will be too hard to implement.

- **Testing**

Throughout the development cycle, you must constantly search for flaws that will lead to unbeatable strategies. If one team is inherently superior or always has an insuperable advantage, your game is out of balance. Similarly, if nothing can defend against a particular unit or weapon, everyone will use it to the exclusion of all others. This quickly ceases to be a fair game, and the player will lose interest and move on to something else.

Lecture 6 - Genre-Specific Game Design Issues II

Simulations

Wish Fulfillment

Of all the genres, simulations are the purest examples of wish fulfillment. Your goal is to fulfill the player’s fantasy of doing things he can’t do in real life.

The catch is that players have different fantasies, and you must decide which ones your game will fulfill. One player might consider himself a student of mechanical physics and approach the sim as an exercise in the precise re-creation of reality. Another might be looking only for the adrenaline-filled rush of operating high-performance machinery in demanding situations.

Hard-Core versus Casual

After you decide on your target gamer, tune the reality of your game accordingly.

For the hard-core, the physics model must be accurate. Measurements must be precise. The controls must respond as they would in real life.

The casual gamer, however, wants to “get in and go.” He doesn’t want to be bothered with learning a million controls before he can *do* something. For casual sims, you must simplify the controls and simulate the fantasy the gamer has in his head, instead of the reality.

Remember: The serious gamer wants real life; the casual gamer wants it to be like the movies.

Interface

The more functionality you put in the hands of the gamer, the more complex the interface becomes. Even if you’re designing a hard-core sim, keep the interface as simple as possible. Hide any game controls that don’t have to be on the screen by nesting them inside menus, rather than clutter the basic view with too many options.

Keep It Fun

Whether serious or casual, your sim can’t afford to be boring. The player doesn’t just want to operate the machinery; he wants to use it in pursuit of some goal.

Sports Games

Some people can’t get enough of their favorite sport. They play it, they watch it on TV, they form fantasy leagues, and they buy videogames as well. These people are *fans* in the original sense of the word—*fanatics*.

Know the Rules

First and foremost in a sports game, you must get the rules right. Keep the official rulebook by your side as you design the game.

Knowing the rules doesn’t mean that you can’t let the player change them! It’s always a good idea to let him customize the game to suit himself.

The Meta-Game

Is your gamer going to be an athlete competing in a single event?
Is he the coach of a single game?
Is he a manager trying to win a season of games?
Is he an owner trying to build a franchise?

Each of these activities results in a different game, and the user needs to know by looking at the box which of these games you have designed.

The Look

Sports games lead all other genres in the realistic depiction of human bodies in motion. Here is where you find the best animation in the business.

Make sure your art director and tech lead are comfortable with this technology before moving forward.

Managing the camera is also complicated. In some games all the action is in one spot. In other games, the action is spread out over a much wider area, especially when team formation is important, such as in football.

Features and Interface

Select the most important features, and then tune and polish those, rather than including everything under the sun without having time to make sure that they all work.

Sit down with the controller for the game system, and think carefully about how its button layout matches the kinds of actions the user will have to emulate on the field.

Fighting Games

Fighting games are simple and direct, yet they can be very engaging. This is one of the few genres to assume that the players are physically sitting side by side and can talk to (and taunt) each other. Your goal is to create quick bursts of swift and intense action, followed by more of the same.

Because the focus is so tight, great graphics are a must. The only things players see are a confined fighting area, a relatively static backdrop, and the two fighters. These characters are the most visually developed of all the genres, because the processor can focus so much attention on them.

Each of the characters must have a unique look that conveys personality, and a set of distinctive moves that are interesting to watch. The animations must be perfect.

The characters and moves must be well balanced. If one character has unstoppable moves, everyone will want to play him. If another is too weak, no players will choose him as their avatar. Either case would be evidence of poor design.

Pay attention to weapons, special graphics effects and sound effects, because they add a large portion of the flash and dazzle of the games.

Manage the damage points so that the rounds are neither too short nor too long.

Start the player with a set of easy-to-learn moves. The button-presses should do approximately the same things for all the characters (for example, High Attack, Low Attack, Defend, and so on), but you should also build in special moves for each character that the players can learn as they slowly master the game.

Casual Games

Your casual game should be “a mile wide and a foot deep.” It should be easy to learn, but *not* difficult to master. Although the gamers’ game generally offers increasingly difficult levels that require more profound strategic decisions or enhanced levels of skill, your casual game should simply provide “more of the same” to achieve additional hours of gameplay. This means that you must design a simple, uncluttered interface that presents the player’s choices in an easy-to-understand format.

If the game is an adaptation of a real-world game such as Hearts, Solitaire, or Poker, you must also remain faithful to the rules. However, give the player the option to customize those rules to his own liking.

Remember that a casual game is often played in short bursts, during lunch hour or a 15-minute break. The player wants to get in, have some quick fun, and get out. You cannot require him to retain information from session to session. He wants to start each time with a clean slate, although it’s also good to provide features such as High Score and Best Time.

God Games

God games have no preset “win” condition. Thus, they are sometimes called *software toys* or *sandbox games*. You must still design a compelling activity that is fun for the player, but instead of pushing him in a given direction and telling him when he has finished, you let him choose his own path.

Here are some design keys:

- Give the player a huge variety of interesting building blocks to play with—especially if it’s a world-building game.
- Design a simple interface, and let him zoom in to concentrate on small details and zoom out to get the bigger picture.
- The interface should also allow him to jump quickly from one part of the world to another.
- The graphics should allow the player to distinguish between units at a glance, and should also convey status information.
- Give the player instant feedback to let him know that his commands have been understood and that the game is working on carrying out his orders.
- If it is a real-time game, let the player vary the speed of events, and allow him to pause the game.

Educational Games

The goal of an educational game is to teach a specific body of knowledge. You must have a clear idea of what this knowledge is from the start. You cannot create a game first and then tack on some educational value at the end. This usually means working with a subject matter expert and adhering to the following guidelines:

- Have a clear goal.
- Consult *state frameworks*, the documents published by state governments that contain the objectives for a given curriculum.
- Target age is important.
- Interactivity, important in every genre, is even more vital in children’s games.
- Keep the interface simple. Don’t clutter up the screen or give the player too many options at once. Make buttons large and easy to click.

- Reward the player often, not necessarily with points but with responses that encourage him to carry on. Deemphasize failure with encouraging words and a hint to push him in the right direction.
- Do steer away from violence, however. It will never be accepted in a children’s game.

Puzzle Games

Many games include puzzle elements, but some games consist *solely* of puzzles, presented on their own without surrounding story or action. The challenge in writing such a game lies not in designing a single brain teaser, but in creating enough puzzles to be interesting to a wide range of players and in sequencing those puzzles properly.

Design the kinds of puzzles that you yourself like to solve (verbal, visual, or both).

Remember that what seems simple to you might be hard to the player. When you know the trick to a puzzle, it always seems trivial. However, when you’re on the other side of the great divide from the “A-ha!” experience, even the simplest of problems can seem monumental.

Your goal is *not* to make the player feel stupid. Your goal is to allow him to challenge himself, and to help him win.

Make a graded series of hints available, and always give him the option simply to learn the solution, especially if solving this puzzle stands in the way of gaining access to other puzzles.

Most puzzles should be presented without time pressure, and it is enormously important that the rules be clear. The interface should be simple and allow for trial and error without penalty, by making it easy to reset the problem or undo a particular move.

For the fanatic portion of your audience, go ahead and include some extraordinarily difficult puzzles, but make it clear to the player that he is entering “expert” territory. There is a real sense of accomplishment in solving very difficult puzzles, and every game should include at least a few.

There are five main genres of puzzle games:

- **Action-puzzle games**. Puzzle-solving combined with time pressure. When designing these games, you must keep the controls simple and get the rhythm right.

- **Story-puzzle games**. Puzzle-solving combined with storytelling. For these games, your main challenge will be integrating the puzzles smoothly into the story.
- **Competitive puzzle games.** Tips for designing these games include: Keep the question format simple.
- **Construction puzzle games.** Here, your main challenge is to create an economical set of pieces that combine with each other in rich, unexpected ways.
- **Pure puzzle games**

Lecture 7 - Level Design

Concept Work

When you begin to think about your level, you must consider:

- Why it is there?
- What function does it fulfill in the grander scheme of things?
- Is it meant to introduce a new character, weapon, or monster?
- How does it move the story along?
- Does it come early in the game, or late?
- Is it part of the single-player game, or is it a multiplayer level?

With these considerations in place, it is time to select a single idea as the focus of your level. This can be a *gameplay* idea

Although each level has a single focus, be sure to provide variety from level to level.

Sketch out your level ahead of time. Experiment with different ways to make the core idea work. Do this on paper, rather than on the computer, so you can cycle through more iterations.

With sketch in hand, you are just about ready to start . . . but not quite. To do a professional job, you need specialized tools. At the very least, you need a level editor. Your editor should provide multiple views, including the player view.

Building the Level

Single-player levels tend to be linear. If the level is too open, the player doesn’t know which way to go and can become lost. You should design these levels with a flow that leads the player along until he has reached his goal.

The architecture should be simple and easy to navigate. The player should be able to learn the map quickly and thereafter never be confused about where he is. These levels should have no safe territory where a player can hide out indefinitely.

Capture the Flag levels should be balanced, with each team’s home base equally easy to attack and defend. Give special thought to color schemes to help the players know when they are entering enemy territory.

The look of the level should be internally consistent. Don’t mix graphical styles within a level, particularly if it is a small map. Although larger maps can contain a series of smaller locations that look different, the style should be consistent within the boundaries of each location.

You should also create visually distinctive landmarks that help orient the player as he navigates through the world. It is easy for players to become lost, and it's part of your job to make sure that they always know where they are.

Gameplay

Goals

Give the player a goal.

In too many levels, the player knows only that he needs to keep moving and shooting until the magic word *Loading* appears, signaling that he has somehow accomplished whatever the level designer had in mind.

Make sure that the player knows what his objectives are for each level. You can do this either in a cutscene prior to the level, or within gameplay when the mission gets under way. It is also a good idea to create a screen the player can always access to get his current status and a simple restatement of his mission.

Mission briefings can be presented in whatever manner fits your fiction. In a fantasy game, a ghostly apparition of a wizard can give the player a quest. In a military game, the commanding officer can issue a set of orders and objectives. Just make sure that when the player enters the level, he knows his goal. Do not let him stumble along saving, shooting, dying, and restoring, until he walks through a door that looks like any other and learns that he is done.

Also, let the player know where he stands in relation to his goal. If his success depends on preventing the enemy from reaching some target, tell him how his opponent is doing.

Structure and Progression

Ease the player into the level, and build up the difficulty as you go along. Do not make the hardest part of the level the first thing the player has to do.

The structure and progression of a level mirrors that of the game itself. Thus, you should build conflict in a series of ascending arcs until you reach the climax. Give hints and teases of what is to come. Create a sequence of ever-tougher challenges, interspaced with bits of humor, and include breaks in the action so that the player can collect his wits before moving to the next challenge.

Do not just keep throwing things at the player. Vary the pace. There should be times when he's frantically trying to stay alive, other times when he's warily exploring, and still other times when he's safe and able to absorb the information he's gathered.

Flow Control

Two hidden problems of single-player level design are how to contain the player in a given area of a level until he has accomplished what you want him to and how to prevent him from returning to that area after he's done.

The first problem is especially prevalent in open-air levels, where there are no natural barriers to keep the player from moving around.

There are many solutions here; the key is to be aware of the problem. You can create natural barriers that are destroyed as a byproduct of the player making progress in the level.

Or you can provide naturally occurring choke-points in the level geometry that are guarded by mini-boss monsters. The point is to ensure that the player doesn't get too far too fast.

The second flow control problem is how to close off an area after the player is done with it. There are many reasons for doing this, including better memory management. If a player cannot return to a given area of the level, he knows that he's making progress and doesn't wonder whether he left any tasks unfinished back there. Of course, closing off the area also means that the programmers can free up the memory associated with making everything there work.

Again, there are many ways to accomplish this after you become aware that it needs to be done. The simplest solution is to create a one-way barrier that blocks the player from going back after he has crossed it. This barrier can be anything—a door that locks when the player goes through it, a waterfall that the player goes over but cannot climb back up, or perhaps a narrow pass that is blocked by an avalanche after the player passes through.

Degree of Difficulty

A single-player level should never be so hard that the player keeps dying again and again.

Challenge is good—even a little frustration is good—but do not make it so tough that only experts can survive. It is fine to design for the hardcore gamer market, but if the average player can't have fun, you won't stay in business long.

How do you satisfy both the expert and the novice? Build multiple types of challenges into the level.

As a level designer, bring everyone through the same front door of your level, ensure that there's enough of each kind of challenge, provide good signposts so that no one gets lost, and make certain that everyone finds his way to the exit when the day is done.

You should also design multiple ways for the player to beat your level. There shouldn’t be just one strategy the player must follow to succeed. The expert can choose a high-risk, high-reward strategy, and the novice can plod along on a safer path, but each player will succeed in his own way and be satisfied.

Balance

There should be neither too many nor too few resources in a level. The player should be concerned about running out of ammo or nervous that his health is running low, but there shouldn’t be so little of either that he spends all his time hunting for them. Ideally, the resource should appear just before he starts to panic. Divining when that moment occurs is part of the art of the level designer.

Balance risk and reward. The more dangerous the weapon or valuable the armor, the harder you should make it for the player to acquire. In multiplayer games, the more powerful a location, the harder it should be to get there. The best weapons should spawn in places where other players have a shot at whoever tries to go for them.

In multiplayer games such as Capture the Flag, each team should have a roughly equal chance of prevailing. The weapons should be evenly distributed, the bases should be equally defensible, and the most powerful weapons and power-ups should be equidistant from the bases so that each team has the same opportunity to get at them.

Puzzles

The single biggest problem of puzzles in action games is that players often do not know where they are. Sometimes a player will slay all the monsters on a level, yet still be left wandering around wondering why he’s not done. After a while, he’ll either give up in disgust or reluctantly reach for the strategy guide.

This problem arises from the wealth of realistic graphical detail we can now put in our levels. If we can make the entire level look interesting and beautiful, how do we draw the player’s attention to the spots that are actually important?

The answer is to give the player clues, to leave a trail of breadcrumbs that he can follow, instead of bashing his head against the level in an orgy of trial and error. The size of the breadcrumbs determines the difficulty of the puzzle.

Other Design Tips

Here are some other tips to keep in mind as you design your level.

- Avoid head fakes. If you have something in a level that looks important, and the player expends considerable effort on it, if you put in a big red door with tough guards on either side and gun turrets along the passage that leads to it, the player had better find something of value on the other side of that door, not just an empty room.
- Work with your AI programmers to learn the capabilities of the AI. Design your levels to take advantage of the AIs' strengths and hide their weaknesses.
- Asset sequencing. Your level is part of an overall progression. You need to know which resources the player has when he begins your level, and which new assets you will be allowed to introduce.
- If you need to deprive the player of a tool or weapon he's previously acquired but that will break your level, go ahead and take it away from him, but make sure that you have a good game reason for doing so. A common example of this is when a player is captured and stripped of all his weapons, and his goal in the level is to escape.
- Do not design a level where the player has to use every weapon in the game to get through it.
- Accommodate different playing styles. Some people are cautious. They take two steps and then pause to look around in all directions before proceeding two more steps. They're fearful of missing something and fearful of failing. Others speed ahead, figuring that they can deal with anything that shows up. The sooner they land in a nest of bad guys, the sooner they can have some fun!
- Provide eye candy, but do not reveal everything all at once. Hold things back so that the player continually has new and interesting things to look at throughout the level.

Evaluation

- Level design is an iterative process. No one conceives a level in his head, builds it once, and then walks away. A level must be honed and balanced constantly, from the moment you start working on it to the moment the producer rips it out of your hands.
- The evaluation process begins with you. You yourself must play your level as you build it, not only to find bugs but also to create new gameplay. As you come at what you have created from the point of view of a player, you say to yourself, "Okay, what if I tried *this*?"

- Be knowledgeable and competitive. Keep track of what other LDs within your company are doing, and go on the Internet to find out what LDs at other companies are doing.
- After you have the rough level in place, let others come in and look at it before you try to polish it. Listen to their comments about the basic ideas. Accept suggestions that solve problems and amplify the theme. Reject suggestions, even if they are interesting, that confuse the player, detract from the theme, or upset the balance.
- Remember, there’s nothing new under the sun, and anything you create could remind *someone* of *something*. If you let this get to you, you will be paralyzed. Ignore it. Go ahead and create.
- Draw on the work of others for inspiration, but process everything through your own filter. Be as original as you can.
- Change the geometry in the tight spot where the player’s collision cylinder becomes trapped. Make sure that all the holes have been eliminated. Find the places where the frame rate is bogging down, and analyze your engine’s capabilities to eliminate the cause of the slowdown.
- After you’ve polished the level, open it up to the testers again. Watch others play your levels, but put some tape over your mouth. Your job is to observe and learn. Rely only on the word of fresh testers who have never seen the level before.
- Know when to walk away. Levels are never perfect. You can always tweak something to make it better. You must develop an internal barometer that tells you when you are done, and you must be able to listen to outsiders as well.

Lecture 8 - Math and Logic in Games

Even though a designer doesn't need to be proficient in advanced math subjects, it does help to have a basic understanding of them in order to converse knowingly with the programming team during production. It is also important to have a good sense of logic, especially Boolean logic, which is used in scripting languages to change conditions on the playfield during play.

Probability and Statistics

One branch of mathematics a designer would be wise to study is probability and statistics. Many game functions involve some kind of random number generation, and a designer should understand the basics of probability─how likely it is that a given result will happen. In some games where multiple random outcomes may result from one game mechanic─for example, determining a critical hit in a role-playing game─understanding the probabilities for possible results is important to successfully balancing the game.

Statistics, which can be used to analyze results after they happen, is also an important design tool for balancing game play. If testing reports that a critical hit against enemies happens multiple times per combat encounter, then the probability for a critical hit is likely to be too high. During the initial design process a designer can use a paper prototype to test some of the more important game mechanics to see if they are within the boundaries he imagined.

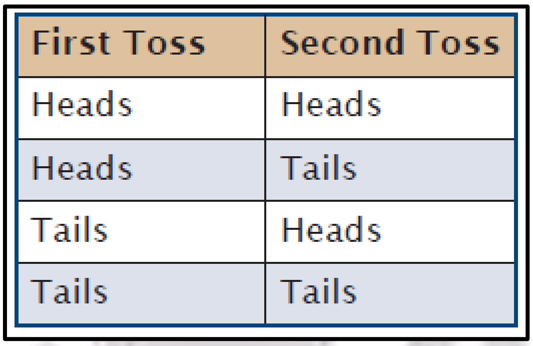
Coin Flipping

The simplest probability is the flip of a coin, where there are only two possible results (ignoring the probability of the coin landing on its edge)─heads or tails. There is a 50% chance of either result with a flip. If the coin is flipped twice, there is still a 50% chance that the result will be heads or tails. However, to see how often either two heads or two tails come up, a simple chart can be created with the possible results (see Table 1).

There is a 25% chance for any of these results. However, the likelihood of a coin coming up twice either heads or tails is only half that of getting one heads and one tails. That is, there is a 50% chance of tossing a heads and tails and only a 25% chance of tossing either two heads or two tails.

Each time an extra flip is added, the results are to the next power of two: 1 flip = 2^1 (2 results), 2 flips = 2^2 (2 × 2 or 4 results), 3 flips = 2^3 (2 × 2 × 2 or 8 results), 4 flips = 2^4 (2 × 2 × 2 × 2 or 16 results), and so on.

Table 1: Results for two coin flips.



First Toss	Second Toss
Heads	Heads
Heads	Tails
Tails	Heads
Tails	Tails

Randomization in Games

Most games include some kind of randomization to add uncertainty to outcomes. A few games like chess and checkers do not rely on randomization, and winning depends completely on the skills of the players. Such games are said to be *perfect information games* because nothing is hidden from the players who know exactly what can happen from turn to turn. There is no hidden information (for example, cards being dealt face down) and no randomization in such games.

Most video games rely on some randomization, which not only can keep the players guessing about what will happen next but also allows games to be replayed because each play through will be different, even though the starting conditions are always the same. Such games are *imperfect information games* because the players cannot predict the outcome of game actions that use randomization. *Tetris* is an example of an imperfect information game because the player doesn’t know the order in which the pieces will appear at the top of the screen. There are seven pieces in the game, and they appear randomly, forcing the player to figure out quickly where to place them.

Random Number Generators

Pseudo-random number generators–algorithms that generated long series of random numbers whose values are determined by a fixed value called a *seed*. The problem with this approach is that eventually sequences of numbers start repeating or the algorithm uses too much memory that is needed elsewhere. The algorithms for generating pseudo-random numbers have grown more sophisticated over time, so it becomes less likely that player will start detecting patterns they can react to.

One tool that can prove extremely helpful during game production is a text-capture program that loads the outcomes of the random number generation algorithm into a text file. While the designers could try to analyze the outcomes visually, it would help to have another tool created that searched for patterns in the text files.

As long as there are no glaring errors or repeated sequences found, it is unlikely players will notice anything wrong with the game. If problems are detected, the programmers may want to try a different algorithm.

Using Dice for Randomizing

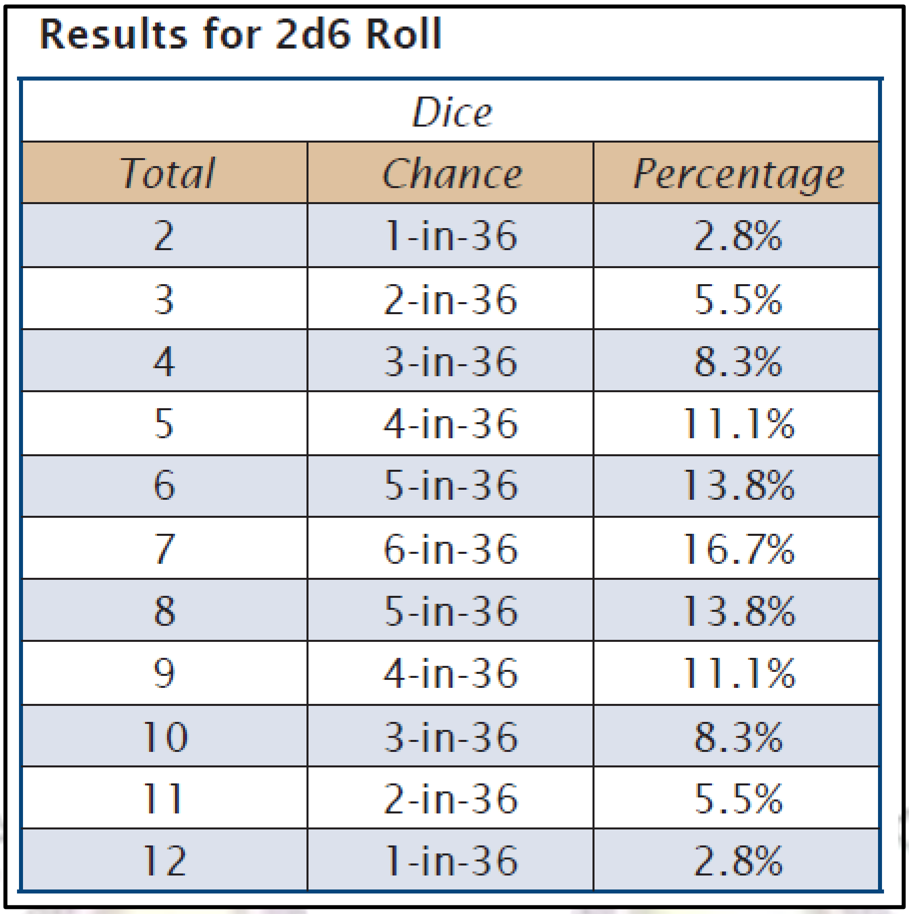
In board games, randomization can take several forms. Players might need to use a spinner to generate a random number or the draw of a card from a deck can make each turn different. The way most commonly associated with board games is to roll a die (or two or more) to generate random numbers, which are then used to determine movement, combat, or other game actions.

Some games use a single six-sided die (referred to in game parlance as d6) to generate a number. In this case, there is a one-in-six chance (16.6%) that any number will be rolled. Using only a single die offers a limited number of results and any result is equally likely.

Most games tend to use two six-sided dice (referred to as 2d6) to get higher numbers and a different probability of results. Unlike a single die, where all results are equal, when rolling two dice, the results can vary greatly. There are 36 possible results for rolling two dice. Note that there is no “1” result since the minimal number of each die is “1” and when added together, they equal “2”. Table 2 shows the possible percentages of 2d6 dice rolls.

It is interesting to note that a double result is as likely to occur when a ”7” is thrown (16.7% in both cases). Using two dice is popular in games, especially gambling games. The results are weighted towards the middle (results of 6, 7, and 8), and players roll extreme results (2 and 12) rarely. On a board with around 40 spaces, such as the *Monopoly* board, the median results mean that tokens move at a nice pace, with only an occasional burst of speed or minimal movement.

Table 2: This chart shows the chances and percentages for two six-sided dice rolls



Results for 2d6 Roll

	Dice	
Total	Chance	Percentage
2	1-in-36	2.8%
3	2-in-36	5.5%
4	3-in-36	8.3%
5	4-in-36	11.1%
6	5-in-36	13.8%
7	6-in-36	16.7%
8	5-in-36	13.8%
9	4-in-36	11.1%
10	3-in-36	8.3%
11	2-in-36	5.5%
12	1-in-36	2.8%

Another Way to Look at Dice Rolls

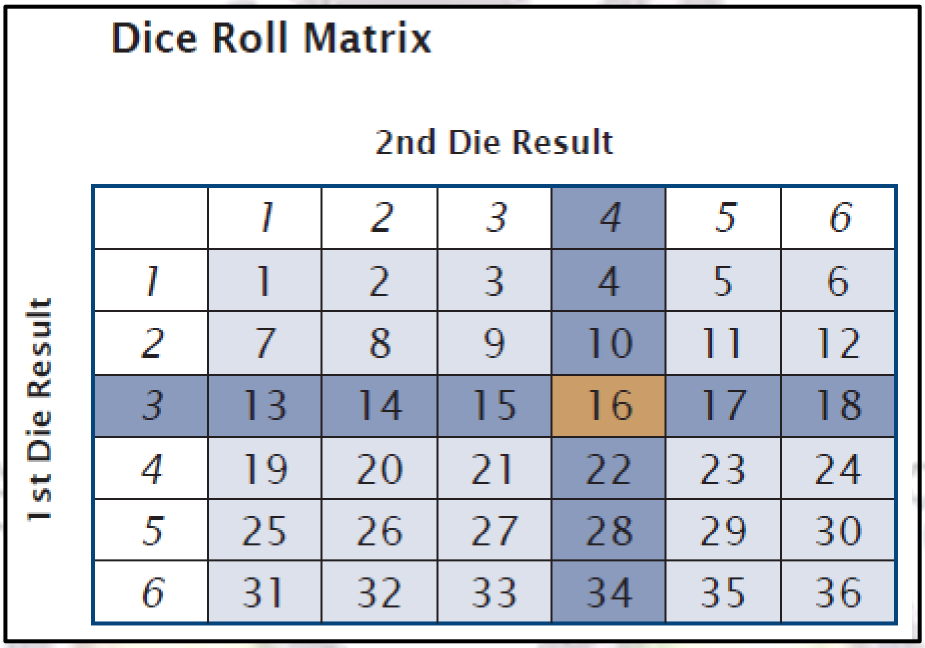
So far, there are some disadvantages of using six-sided dice in games. Rolling more than one die and adding the numbers together cumulatively means that the lowest result can never be a “1” for two dice, “2” for three dice, “3” for four dice, and so on.

One way is to throw the dice and deal with the results sequentially instead of cumulatively. For example, in a game a player has the option to throw two dice (2d6) one at a time. He throws the first die and considers the result before deciding to throw the second die.

Maybe using the first die result lets him move his token to a space with a positive result, so he decides to use this result and ignore the second die altogether. Or, if he has to throw the second die, he then moves his token that number of spaces after moving it a first time–in effect, having two movements in the same turn. A similar approach is used in Backgammon, where a player can split the results of the dice roll between two pieces or use the combined result to move one piece.

Another approach is using a matrix for results. In this case, the row corresponding to the first die roll is cross-referenced with the column corresponding to the second die roll. In this case, there are 36 potential outcomes and there is an equal chance of obtaining any result. For example, in Table 3, the first roll of a “3” is cross-referenced with the second of “4” to get the result of 16. Matrices are used in some games, particularly for resolving combat in paper wargames.

Table 3: This matrix cross-references the result of two six-sided dice being



Dice Roll Matrix

2nd Die Result

1st Die Result

	1	2	3	4	5	6
1	1	2	3	4	5	6
2	7	8	9	10	11	12
3	13	14	15	16	17	18
4	19	20	21	22	23	24
5	25	26	27	28	29	30
6	31	32	33	34	35	36

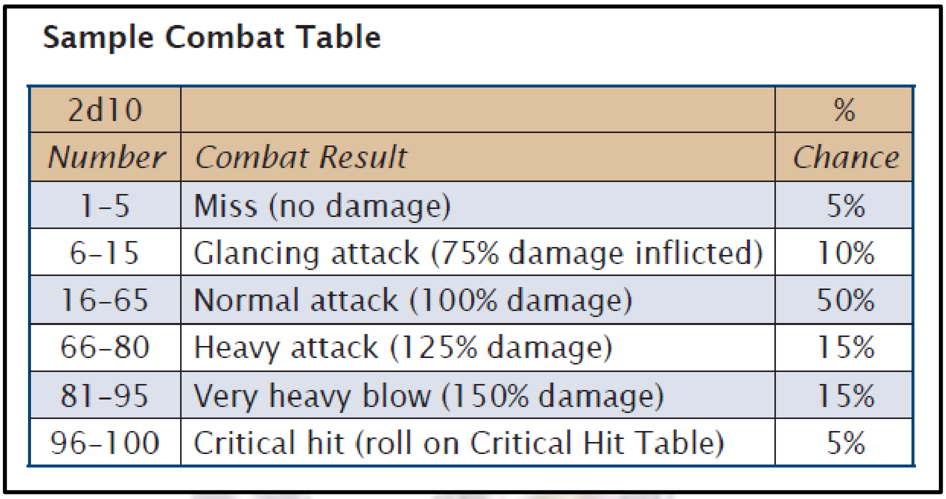
Percentages

Percentage dice had been used earlier but were mostly 20-sided, requiring players to divide dice results by “2” to get a percentage result of 1‐100. The advantage of such a ten-sided die (referred to as d10) was that by rolling two dice in sequential order it was possible to generate 100 numbers.

One die was assigned to roll for tens and the other for single digits, giving a range of 1‐100 or 00‐99 (the latter configuration is more often used in electronic game design). The result of 00 on both dice can stands for either 00 or 100. Two ten-sided dice are referred to as percentile dice.

For example, if the designer wants one game action to occur most frequently, he can assign a range of 1‐50 for that action. Less frequent actions can then be assigned smaller increments down to a single die roll result, which would make that result very rare (1-in-100 or 1% chance). Thus, a percentile combat table for a role-playing game might looks something like Table 4.

Table 4



Sample Combat Table

2d10		%
Number	*Combat Result*	*Chance*
1-5	Miss (no damage)	5%
6-15	Glancing attack (75% damage inflicted)	10%
16-65	Normal attack (100% damage)	50%
66-80	Heavy attack (125% damage)	15%
81-95	Very heavy blow (150% damage)	15%
96-100	Critical hit (roll on Critical Hit Table)	5%

Percentages in Video Games

In video games the resolution of game actions is handled by the game logic. Any range of randomized possibilities can be handled equally well, so the designer is free to use whatever range best suits each game action. An advantage of creating a paper prototype of the game before coding begins is that the designer can test various probability ranges for game actions and make as many changes as necessary before settling on the most workable ranges.

If it feels more comfortable working with probability ranges of 1-100, the designer certainly can do so. When the programming team creates the algorithms that resolve the action, they can either keep the designer's desired ranges or change them as necessary to simplify the code.

Keeping the Math Simple

Many complex games use numbers of some kind to present important information to players–for example, how many experience points are needed to get to the next level in an RPG. In most cases, the numbers should be relatively low and easy to comprehend at a glance. If the numbers presented to the players get up into the thousands or millions, it takes time for players to parse them to figure out what is happening and what to do next. It is also difficult for the player to process too many numbers at once, so having a whole interface screen filled with many different number fields can look more like an Excel spreadsheet than a screen from a game. Likewise, as more video RPGs move to real-time combat action (instead of turn-based), it can be confusing for the player to see a slew of numbers fly around the screen as attacks are resolved.

If the numbers appear on secondary screens that aren’t used much during play, then they can be large since the player has plenty of time to determine how many tens or hundreds of thousands of experience points are needed to reach the next level.

There is a trend in many games to cut back on numbers the player has to deal with and replace them with other visual clues. An icon near the character that grows progressively redder as the player takes more damage in combat can replace the slew of numbers flying across the screen as each attack is resolved. A countdown timer can be used to show how much time remains until an action happens.