COMP 550 Study guide

Francis Piché

September 18, 2019

Contents

Part I
Preliminaries

1 License Information

These notes are curated from Professor Jackie CK Cheung lectures at McGill University. They are for study purposes only. They are not to be used for monetary gain.

Part II
Games in General

2 Study of Games

In the past, the study of games was mostly focused on the mathematical aspects. Combinatorial games tended to play well into this way of thinking as they were simple to model. These games are typically:

- Two players
- Turn based
- All player have full information (no random events)

Games like chess fall into this category, and have been studied extensively.

Modern games on the other hand, have added a few layers of complexity. Things tend to be real time, graphical, and the emphasis is less on the study of the games, but more on the software engineering aspects of producing them. The industry focus has been on how to make them more enjoyable and deliver at scale.

Nonetheless the common areas of study in modern games include:

- Graphics and animation
- AI (automating components, non-player characters, etc.) Note: the goal is not necessarily to have a "smart" AI, but rather one that is fun to play with/against. An AI that always wins is not fun...

- Real-time physics and simulation
- Narratives
- Interfaces, ergonomics and haptics

The general trend in games is not necessarily "correctness" or realism, but more how to balance correctness and efficiency in a way that is fun.

2.1 What is a Game

"A game is a series of interesting choices - Sid Meier.

The definition of a game is not well defined, but we can try to find common elements of games in a general way:

- Limits: Games have some system of rules, if not explicitly defined then implicitly limitations are imposed through the artificial context in which the game is played. Games are generally played in some abstract separate space from the real world in which the rules are "different".
- Goal: Games generally have some goal, even if it is not well defined. There is some sense of progression or achievement.
- Challenge: Instant wins are not fun. Games tend to have some layer of difficulty to make them "winnable"

2.2 Game Mechanics

A game mechanic, whether composite or elementary, is the "essence" of a game. As opposed to rules, "what you could do", mechanics are "the means by which you do". For example, in a stealth game you might have a "hide in the shadow" mechanic, while the rules are that you must not be spotted.

2.3 Genres

Ludic games such as Tetris focus on gameplay and mechanics as opposed to narrative games which focus on telling stories. Typically games sit somewhere in a pyramidal shape between ludic, narrative, chance and simulation.

2.4 Fun

Why people play games goes back to the greeks who categorized 4 main reasons:

- Agon: Competition

- Alea: Chance
- Mimicry: Simulation / role-playing
- Ilinx: Vertigo, momentum, physical stimulus

More than this, games rely heavily on **the illusion of winnability** to stay entertaining. That is, the idea of ensuring that the goal is not immediately achieved, but that the player is presented with a clear reason for their initial failure, and that reason must be *correctable*. This ensures the players motivation to try again, and a sense of progression.

2.5 Flow

Flow is the state a person enters when doing certain activities in which:

- There is a task to be completed
- The person has an ability to concentrate
- There are clear goals
- There is frequent feedback
- Sense of control over ones actions

The consenquences (flow) are:

- Deep, effortless involvement
- Reduced sense of self (hunger, discomfort ignored)
- Time distortion

2.6 Immersion

Also important to games is their sense of "immersion". This can come from "sensory immersion", in which sound, visuals and occasionally haptic feedbacks combine to give a sense that the player is within the game world.

Another form of immersion can come from challenge. Attempting to overcome something difficult often leads the player into a state of immersion.

Imaginative immersion takes place in the players mind, rather than as an external stimulus. Puzzles are a form of imaginative immersion since most of the game is played in the players mind.

2.7 Player Types

Not everyone likes the same things. The guy who invented MUDS (Multiplayer Underground Dungeons) observed 4 different player types in his games:

- Spades (Explorers) who try to do everything, see everything and explore the entirety of the game.
- Hearts (Socializers) who focus on interactions with other players.
- Diamonds (Achievers) who try to "win" the game by maximizing score, resources etc.
- Clubs (Killers) who get their kicks by killing other players or generally hindering others who are playing the game.

3 Narratives

Games are by nature interactive, and so the narratives we'll focus on are interactive narratives in which the choices of the player have some impact on the game.

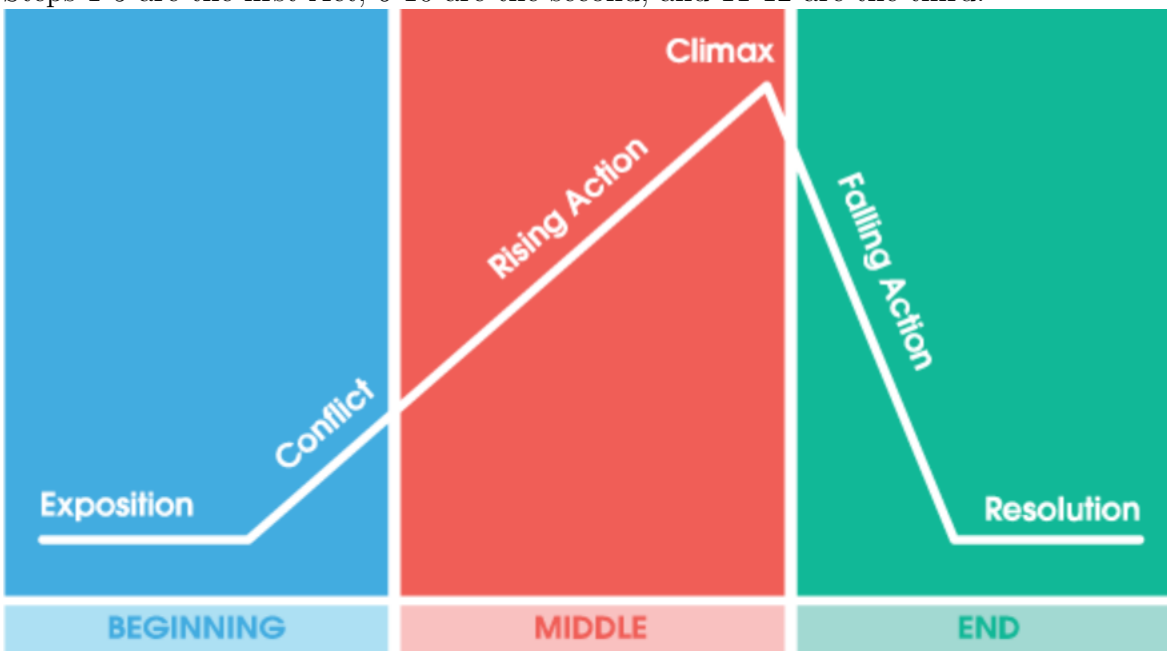
3.1 Plot Structure

After some study, there have been 36 common plots found which occur frequently in stories. Amongst these, the Hero narrative template emerges as being a very common plot structure. This structure consists of 3 Acts.

1. Hero begins in ordinary world with some minor concerns
2. Call to adventure through some external influence / catastrophic event
3. Hero initially reluctant
4. Meeting with a Master (some authority figure that convinces the hero)
5. Crossing of threshold (point of no return in which the world is no longer the same)
6. Tests, enemies, allies and discoveries. (This is the bulk of the story)
7. Approach innermost cave
8. Big fight with nemesis
9. Reward obtained
10. Road Back
11. Resurrection (not done yet must go back for one more trial)

12. Final Return (ordinary world is stable again)

Steps 1-5 are the first Act, 6-10 are the second, and 11-12 are the third.

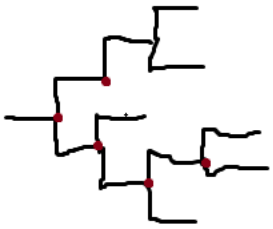


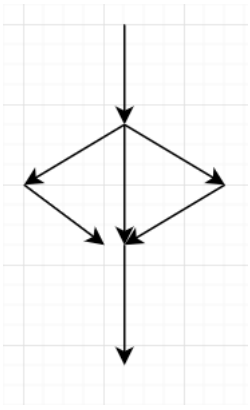
3.2 Interactive Narratives

One of the first forms of interactive narratives came from IF books in which the reader would be prompted with a choice, and each choice would have a corresponding page to turn to.

There needs to be some balance between making enough choices to feel that there is some interactivity, but not too any as to be overwhelming. There also needs to be a sense of impact, but not so much so that decisions are too stressful.

3.2.1 Interactive Narrative Structures

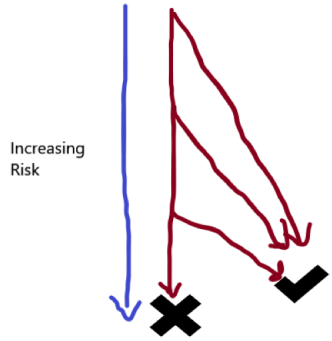


Tree  flow

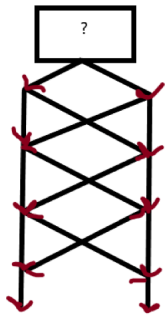
A tree structure allows many different experiences, but requires a lot of work to implement as there are many many disjoint paths a player could take.

The flow chart (DAG) (Bottle-Branch) structure gives more control to the game makers but less to the player.

Danger progression is a form in which the player is given many opportunities to opt-out of a dangerous path as risk increases. This gives the advantage that the player is able to explore risky situations without immediate death:



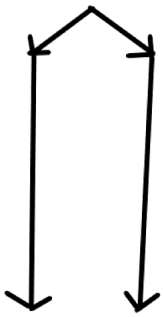
Track switching is a similar idea where the player can "change their mind" at any point:



Gauntlet is similar to Bottle-Branch in that it has one main line with a few inconsequential side-branches that give the player the feeling of options while keeping a central story-line.

Sorting Hat is the idea that a single decision near the beginning determines a "track" in

which the player will play that is disjoint from the other tracks until the very end.



Open Map is the idea of allowing the player to do anything at any time in whichever order they please.

Cyclegrow is the "groundhog day" idea where you repeat the same things and each time a new path opens.

Quantity-based narratives are when some areas of the story are "gated" by saying you must have X quantity of something before the choice is offered. This comes with feasibility constraints since the player must never get "stuck" unable to proceed with the narrative.

3.3 Quantifying narratives

The size of a narrative can be measured by the number of nodes in the graph (events). The complexity can be measured by number of choices or branches in the graph. For the players path, the maximum and minimum length/complexity should be somewhat balanced.

The distance to a "bad" outcome can be used to measure risk. (How many choices are you away from messing up), and similarly the distance to a reward can be used to quantify reward. So balancing risk and reward can be done by checking the distance of the player to the nearest risk and reward.

The impact of choices can be measured by giving scores to nodes, in which the difference between child nodes is the "impact" score of a decision. The parent node can be the average of the children. In this way, we get a recursive definition of how "impactful" each decision is, all the way up the graph.

3.4 Narrative Flaws and Benchmark

Narrative flaws include uncompletable narratives, discontinuities, pointlessness and logical inconsistencies. A classic benchmark of what your narrative should be able to do is the "Cloak of Darkness" narrative. This is a minimal narrative that demonstrates:

- Different locations
- Objects
- State and non-local interactions within the states.

In this narrative, there is a Coat room, a Foyer room, and a dusty Bar room. The player is carrying a Cloak of Darkness. If the player moves directly into the bar, the room is dark and nothing can be seen. However if the player moves into the coatroom and leaves the cloak there, then enters the bar, the bar will be lit up. Once the bar is lit, the player will see a message in dust on the floor. However if te player enters the bar 3 times, the dust will be disrupted and the message will disappear.

This is known as a "benchmark" because it allows room for choice with non-obvious results, a fair "losability" and "winnability" through careful exploration of the options.

Part III
Procedural Methods

4 Mazes

4.1 Types of Mazes

In **Simple / Perfect** mazes, all areas are reachable, there are no cycles and there is exactly one solution. The player enters one place, exits another. This maze is essentially a tree structure, in which the leaves are "dead-ends", and intermediate nodes are the choices.

Braided mazes are like simple mazes but allow cycles, and have no dead-ends. There are possibly many solutions.

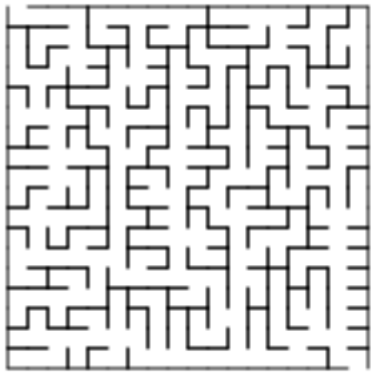
A **Unicursal** maze has only one path and no choices. In a sense it is the "anti-maze", but is a maze none-the-less.

Of course you could mix all of these ideas together for more interesting mazes.

4.2 Building Mazes

Due to the tree structure of mazes, the problem of generating random mazes can be formulated as finding a spanning tree within a grid of nodes.

4.2.1 Prims Algorithm



The above maze was generated using Prims Algorithm, which goes as follows:

1. Choose a root node this is a partial tree T
2. Of all nodes adjacent to nodes in T, randomly select one and add it to T.
3. Stop when all nodes are in T.

This will create a maze with lots of branches, but the algorithm is very simple and effective. We could also turn this into a braided maze by connecting each leaf (dead end) to some nearby node in the tree.

4.2.2 Aldous-Broder Algorithm

This algorithm gives a uniform spanning tree. (One created by adding nodes uniformly at random).

1. Pick a starting point. The "cursor" is here.
2. Choose a random neighbor.
3. If the neighbor is not part of the tree, connect it. Otherwise return to step 2.

At first this algorithm is very fast, but as the graph fills up, it might take a long time to find a node that is not part of the tree yet. Of course you could start start moving the cursor to unchosen nodes near the end but this loses the UST property.

4.2.3 Wilsons Algorithm

This algorithm also gives a UST, and works by adding entire branches at a time.

1. Pick random node outside the tree
2. Wander randomly from that point. If encounter node in the tree,save the path and return to step one. If the random path intersects itself, throw out this path and return to step 1.

This algorithm is very slow at the start, but speeds up as the tree grows. Of course to speed it up you could instead of throw out the whole path when it intersects itself you could save the path, but this loses the UST property if you care about that.

4.2.4 Ellers Algorithm

This creates infinite mazes, by generating one row at a time. It does this by putting nodes in sets that are used to keep track of which columns will be connected.

1. Each cell of the first row belongs to its own set
2. Randomly join adjacent cells only if they are not in the same set. When joining cells, merge the sets into a single set.
3. For each set, randomly create vertical connections down to the next row. (Each set must have at least one connection! (but maybe more than one)).
4. Put remaining cells in the new row into their own sets.
5. Repeat forever.
6. If not infinite, in the last row, join all adjacent cells that don't share a set.

4.2.5 Unicursal Mazes

To make a unicursal maze, you could pick your favorite spanning tree algorithm, and then take the longest path, and throw everything else away, leaving behind a unicursal maze. Or you could walk around the perimeter of a spanning tree, creating a unicursal maze as well.

4.3 Dungeons and Rooms

Mazes don't have rooms, but we can make rooms from mazes by removing dead ends to make "space" for open areas. There are many other algorithms for generating dungeons, but they wont be covered.

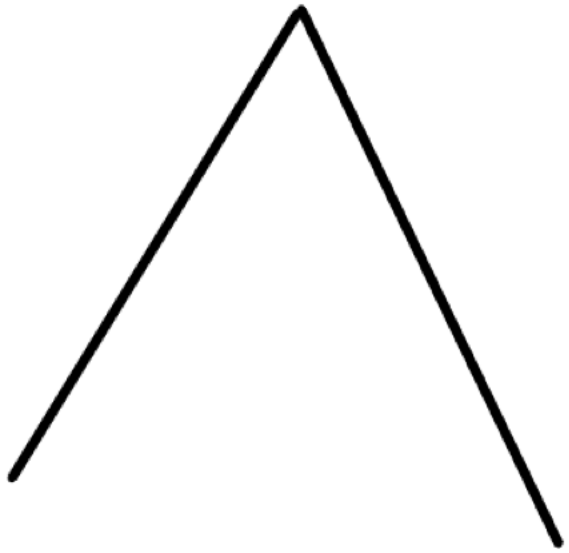
5 Terrains

5.1 Tiles

One approach to procedural terrains is to section-off the world into tiles. Then each tile can have some properties. However this comes with feasibility challenges since adjacent tiles must make sense, and alignment becomes an issue.

5.2 Terrain Generation Algorithms

Suppose we want to create a mountain. In 2D, we can start with the basic shape:



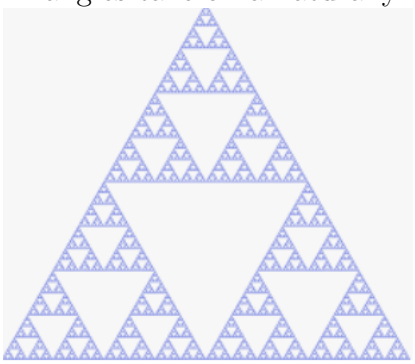
We can't just start displacing points randomly, or it will not look good. We need the idea of having large and small scale displacements in a continuous fashion.

5.2.1 Midpoint Bisection

One way is to take the midpoint of each line, and recursively displace the point along the normal to the line by some random amount. (Positive or negative). The amount by which we displace will need to be proportional to the line-length otherwise the result will not look good.

5.2.2 3D Recursive Triangle

Triangles take on a naturally recursive shape:



So in 3D, could assign a random height value to the corners triangle and recurse. However, each recursion needs to be aware of the others since the triangles have shared edges.

5.2.3 Diamond-Square Algorithm

Starting from a diamond-shape, assume we have some height values for each of the 4 corners. Take the center of the diamond, and assign some average of the corners ± some random displacement.

Next, find the midpoints of each edge. Join this midpoint to the center. Set the height of the midpoint to be the average of the endpoints.

We can now recurse on each newly formed diamond without having to be aware of the others since each endpoint has a well defined height already.

5.2.4 Fault-Line

Another way is to take a square, and repeatedly draw lines across it. For each line, raise one side a random amount and lower the other side some random amount. After many iterations the terrain will appear somewhat bumpy and random.

5.3 Perlin Noise

Theres a lot of good resources about this that explain it better than I ever could... Heres Daniel Shiffman's explanation as part of his "Nature of Code" series: `https://www.youtube.com/watch?v=8ZEMLCnn8v0`