Gamified Course

May 14, 2012

Preface

The following was designed to work with a CS1 or CS2 course. It will hopefully accomadate both majors and non-majors. While an instructor and TA's are recommended, this course could potentially be used as a self-taught online course with little to no modification.

Chapter 1

Introduction

1.1 Fun

- normally a predominant aspect element of gamification
- due to the nature of homework, play is required
- while fun would be nice in order to encourage involvement and effort, it is not strictly required for this design
- that said, fun can be a major piece
  - we are using games elements as an alternative to course work
  - (dr. arne May research gamification pg 4)

1.2 Players

- playing for mastery, not social, fun, or destress (Gamification 20), achievers and explorers, little to no focus on socializers
- retention less important since it is again a requirement

1.3 MDA

- Mechanics
- Dynamics
- Aesthetics

1.4 Concept Exposure

- good variable naming
- project design by example

1.5 Adaptive Challenge

- adaptive AI/pacing equivilent

1.6 Exploration

1.7 Literacy

1.8 Consistent and Constant Reinforcement

Part I

Design

Chapter 2

Topics

All games require a game world and a gamified course is no exception. However, since a player is navigating through knowledge and facts, creating a simple 2d board is not quite enough. Instead, we must navigate through a more conceptual space. Traditionally, a course has a collection of knowledge resources in the form of notes, excercises, assignments, and readings. A student would simply navigate through these materials by picking one up, reading or working a bit, and putting it down. For a game world to be built out of these resources, something must be defined to represent a location as well as a way to move between locations.

Like a room in an interactive text adventure or a square on a chessboard, a topic will be a single atomic location in our game world. However, unlike rooms and squares a topic has no real location in space. However, it is an atomic unit of "knowledge". A topic can be thought of as the smallest unit of fact or knowledge attempting to be taught. Think of it as the smallest subsection within a book; a topic conveys one and only one piece of information.

Conditional Topics

- booleans
- "and" statements
- "or" statements
- using both "and" and "or"
- conditionals (if-then)
- branching (else)
- multiple branches (elseif)

Figure 2.1: Example Topics

The topics contain everything necessary to teach this one concept. This includes tasks, questions, and information. These will be explained more thoroughly in 2.2.

However, it is necessary to be able to group topics. Figure 2.1 shows a few possible topics for conditionals. Although each topic stands by itself, they are clearly related in subject. Subjects will be defined as any term which a given topic relates to. Therefore, each individual topic can have more than one subject. The topic "using both 'and' and 'or"' has the subjects of "boolean logic", "basic coding", "conditionals", "and", "or", and potentially more.

To avoid having to enter every single related subject, subjects can also relate to other subjects. For example, "and" and "or" could both fall under "basic coding" and "boolean logic." In this way, subjects are essentially a form of hiearchical tagging. However, a topic's subject set refers to all explicitly defined subjects for a topic, not any subjects implied by hierarchy.

2.1 Topic Space

Subjects alone are not enough to define how topics relate to each other. There must be a clear way to get from one topic to another. While grabbing a random topic in a subject could work, more likely than not it will result in topics being completely out of context and order. Therefore topics need a better way to define how they connect to one another. We will call the set of these topics and their edges "topic space".

As with sections in a book, there is sometimes a clear order to teach topics. It is also sometimes the case when it makes just as much sense to teach one of many topics next. For this there is a next-topic edge. Next-topic edges are the strongest edges in our topic space, creating the branching paths throughout topic space.

Just as in any game world, topic space has regions. These regions are defined as a set of topics with the same subject-set and their next-topic edges. These regions are analogous to a chapter in a textbook; regions contain multiple related topics and a path through itself. The set of edges creates a directed a cyclic graph within a region; while the path in a region can branch, it can never repeat.

Next-topic edges can also connect to a topic outside of a region. These connections can be thought as a path between regions. This style of edges can create sidebars or an exit for the region.

However, simply traveling to the next-topic does not always make the most sense. Some regions do not fit in naturally in a specific part. This is especially true for sidebar or enrichment subjects. However, even in this case a certain amount of knowledge is required before they can be started. For this, there is the requires edge. A requires edge points to another topic which must be completed before starting the current one. In general, this is the most common way to connect regions.

Figure 2.2 shows a potential subset of topics space. In this diagram, the letters at the start of the topic name refer to the subject the topic is in; "A1" has a subject of "A", "B3" has a subject of "B", and "AB" has a subject of "A" and "B". The color coding defines the region of the topics. As expected, the next-topic

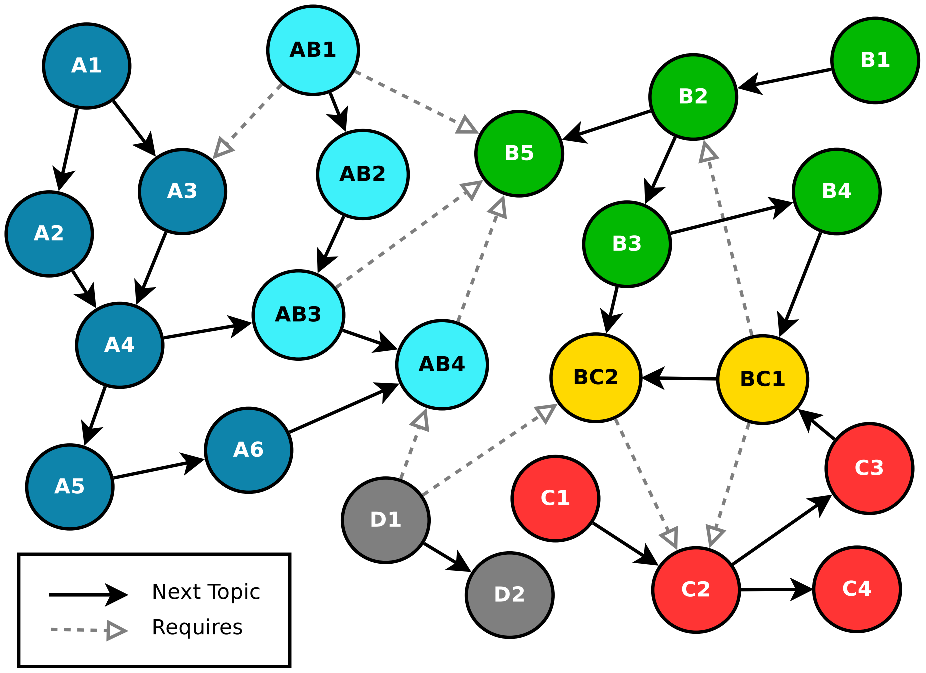

Figure 2.2: Topic Space

edges point to the next topic and requires edges point to the topics that must be completed before the given one. For example, to start topic D1, AB4 and BC2 must be completed.

There are some interesting paths in figure 2.2. For example, a player can take A1 ▷ A2 ▷ A4 or A1 ▷ A3 ▷ A4.

Through next-topic and requires edges, multiple paths through between regions are quickly created. Assuming the player has already completed B5 (the requirement for the entire AB region), the player could get to AB4 in the following ways:

- A1 ▷ A3 . . . AB1 ▷ AB2 ▷ AB3 ▷ AB4
- A1 ▷ A2(A3) ▷ A4 ▷ AB3 ▷ AB4
- A1 ▷ A2(A3) ▷ A4 ▷ A5 ▷ A6 ▷ AB4

These multiple branching paths allow players to create their own way through the structure. Some players will master a subject-set early on and quickly move on to another related region. Others will have trouble grasping the basics of a subject-set and complete many more topics in a given region, but then have to do less examples in a related region.

Note:

- Next-topic does not have to point to a topic where all requirements have already been met
- Next-topic can point outside of it's own region creating a sidebar or exit.
- Next-topic has an implicit requires pointing in the opposite direction. However, while all requires edges must be met, only one next-to edge is necessary.
- Requires glues regions together to create a looser sequence/order.
- Any topic which has no requires edges or next-topic edges pointing at it can be considered an entrance point

Expanding Topic Space This is the simplest definition of topic space. Depending on the requirements of the course there could be any number of addtional topic groupings and edges. For examplem, instead of only allowing a requirement edge between two topics, requires could be attached to a region as well. In figure 2.2, instead of explicitly defining a bunch of requires, the region AB could simply require B5.

2.2 Topic Contents

There is no point in defining topic's without defining what goes inside them; what would be the point of a game that only contained empty rooms? Just as a room can have monsters and items, a topic can contains three types of things: information, questions, and tasks.

2.2.1 Information

Information is the simplest piece of a topic: it is the explanation of a topic. Using the textbook analogy, this is the actual text of a section explaining the material covered. It is not, however, anything having to do with explaining or giving a full example/excercise.

2.2.2 Questions

Questions are simple questions having to do with a topic. These would be the equivilent of what might be found at the end of a textbook chapter or on a pop quiz. Most often they will be multiple choice or short answer. Their one requirement is that they must have a clear correct answer. This excludes essay questions or anything which needs to be graded by a person.

2.2.3 Tasks

Tasks are the actual meat of a topic; tasks are the actual monsters of this game. They are the primary obstacle that a player must overcome to actually complete a given topic. Just as a player may enter a region in an MMO and kill monsters to grind for points or complete a quest, players will enter topic regions and complete tasks.

In a traditional course, a task would be the combination of an excercise and an example. When a player first sees a task, they are presented with a traditional excercise. This would be no different than what would be found on a problemset type of homework or an example out of a text book. Completing this task would award the player a certain number of points (defined in chapter 4.1.) Each task should be about the size of a single word problem.

In addition to just being an problem, the task is also an example. Using something similar to the universal hint system, a player can get incremental amounts of help with a problem. For each task, there are groups of hints. Each group has an initial question, creating a line of questioning each hint in the group answers more about. Groups of hints can lead into one another by linking to another question. As the player requests a hint, cost of the hint is subtracted from the value of the task lowering what it is worth.

Each hint belongs to a line of questioning. This means that various parts of the excercise can have various groups of hints associated with them. In figure2.3, we can see an example of multiple lines of questioning. There is a very general hint which explains what formulas to use and where to find them, and a second line of questioning about how to get the x and y values out of the arguments.

In this way a task is sort of like working through a problem with the option of asking a virtual TA for help. While hints cannot possibly answer every possible question, they can answer the most common ones.

At any point the player can give up on a task and ask for an explanation. The final value of the task is automatically the lowest possible score, usually a single point. This explanation gives the answer and walks through how it was solved.

Because of the option to get the answer, tasks are not able to be lost. This is because topics are a replacement for course materials like a textbook. Just as you cannot lose at reading a chapter, you should not be able to lose at solving a task.

Task: Create a distance function (7pts)

Define a function which takes two points "a" and "b" as arguments and then calculates the distance between them. A point is a tuple containing (x,y).

Hints:

- How do I get started?
  1. *(-1pts)* Use the distance formula: $d = \sqrt{(b_x - a_x)^2 + (b_y - a_y)^2}$
  2. *(-1pts)* You can find the squareroot function in the python math library.
- How do I find b_x?
  1. *(-1pts)* Since the points are tuples, you can get the x and y components out with indexing
  2. *(-2pts)* You can find $b_x$with b[0] and a_y with a[1]

Explanation (1pts): how to write the distance function

```
def distance(a, b):
    return math.sqrt((b[0]-a[0])**2 + (b[1]-a[1])**2)

# you can also unpack the variables
def distance(a, b):
    ax, ay = a
    bx, by = b

    return math.sqrt( (bx - ax)**2 + (by - ay)**2 )
```

Figure 2.3: Example Task

Note:

- The value of a hint can be weighted by how many points it subtracts
- the total points of a task must be at least the total cost of the hints plus the value of the explanation.
- the game state remembers
  - which tasks have been completed
  - which hints have been unlocked and they will always be unlocked
- you get no points for completing the same task twice

Chapter 3

World

It should be noted that the topic space is not exactly the same thing as a game world. Game worlds traditionally also contain some sort of state. Whether this is which doors a player has unlocked or if it is raining outside. However, topic space topic space should only contain information; topic space should not store any information about a player's progress. In a sense, topic space is closer to a map in a video game or a the board of a board game. The game world is a combination of topic space and everything having to do with state (player's progress through topics, current location, etc.)

3.1 State

3.2 Events

3.3 Visibility

- visibility vs locked
- sometimes hiding is just as useful

3.4 Locks

- why?
  - does us no good to have a singular entrance point in topic space
  - especially when dealing with unreleated topics

  - hiding portions
- defining a lock
  - it is simply a set of world metrics
- similar to requires edges in topic space
  - edges are used to define requirements in knowledge
  - these are like facts or physical bounds
  - locks are used to define the structure of the course
  - while not hard and fast requirements, they are there for a specific implementation
- abstracting the concept of requirement
  - permeable
  - multiple levels of locked
  - lock, recommended, suggested, alert
- Using an event model to slowly release locks
  - signal processing
  - pub sub model
  - triggers, events
- multiple locks per item
  - topics are locked
- Possible uses

Chapter 4

Points and Progression

As is, the game world simply stores the player's progress through topic space. There are some number of completed topics, available topics, and which hints are unlocked. This is not a particularly engaging or meaningful way to measure progress. First, not all tasks are worth the same number of points so having a raw number completed does not mean much. Second, there is an overwhelming amount of detail. Games have long since solved this issue. In an RPG, a player doesn't always track the number of quests they completed or the number of monsters they've killed. The only thing the player has to track is their level.

Within this gamified course, the player's progress is grossly measured by their rank and skill levels. The rank gives a broad overview of where the player is within the course. Skills act as a progress bars for the completion of regions in topic space.

4.1 Points

Points are the most basic way to reward a player for their actions. To look at it another way, points are a measure for how much effort a player has put into a game. The primary difference between this game and many others is that points are awarded based on effort rather than skill. As players complete activitities they should be constantly awarded points for what they have done.

Skill Points (pts) The skill system is similar to the one in Elder Scrolls III: Morrowind. As a player performs an action, they slowly gain points and level in a specific task. This means the more locks you pick, the better you are at lock picking. The amount of points granted per lock is directly related to how difficult the lock is to pick.

- awarded for achieving specific goals in assignments, class, and other
- specifically relates to a given topic
- completing a homework assignment boils down to collecting enough knowledge points
- rigidly and predictably rewarded
- marks progress throughout a given assignment

4.1.1 Awarded vs Gaining

Experience Points (xp) Whenever a student does anything, XP should be awarded.

However, the amount of experience awarded dwindles. This should help prevent students from gaming the system by only completing the easy tasks. If no KP then there is no XP awarded.

- proportionate to awarded KP and respective topic level
  - will eventually dwindle to 0
  - attempting a topic above skill level earns more xp, far enough below xp level earns nothing

4.2 Skill Levels

As the player gains more and more skill points...

Specifically, a skill is a measure of progress per individual topic subject. Each skill has a subject, its current level, and progression of how many points are required to move up a level. Skills can optionally include a max level after which no more points can be gained. All actions in topic space award some kind of point. As the player gains more skill points (4.1), they will go up in skill level. Also, just like their subject counterparts, skills also exist in a hiearchy. As points are awarded to a child skill, they propagate up to the partent skill.

As the skill level goes up, the points gained from the same work goes down. For example, imagine a task which awards 10 pts for for looping level 1. Completing this task when the player first starts will award the full 10 pts. However, if the player had a looping level of 3, only a fraction of the points (let's say 5) would be awarded. There could potentially be a point where no skill points are received for a task that is too low level. This is to prevent players from simply solving every single easy task and never moving forward. It creates a maximum number of points that can be awarded per subject so that other subjects have to be worked on.

In addition to acting as a progress bars, skill levels are should be one of the more common keys for requirement locks.

4.3 Rank

A players's rank is more general than individual skill levels. The player rank is more indicative of how much a player has done total. As a player moves up in rank, they will get access to new topic regions and things to work on. Ranks can be seen as milestones for the course.

Ranks are slightly different than traditional experience based leveling. While ranks are predominantly based on experience points, moving up in rank can have other specific requirements. Ranks are closer to a combination of leveling and a technology tree; once the player has enough experience points and meets certain requirements they move up in rank. A player may be required to complete a certain number of tasks, a project, have a certain skill level, or even just wait for a specific date to pass.Ranks use requirement locks the same way as topic space.

As the student moves through the ranks, the game world changes. New topic regions may unlock while simplier ones may be reorganized and regrouped. The student's skill list can be modified to reflect this new rank. For example, skills in CONDITIONALS and ITERATION may be grouped into a larger BASIC CODING.

The overall goal if this course is being taught is to try to have every student moving through ranks at about the same rate. If the entire class is ahead or behind the expected rank, this is a quick indication that the pace of the material might need adjusting. Alternatively, a student's rank can quickly show if they are falling significantly behind the rest of the class.

An effort should be made to make the amount of time between ranks consistent. This is to regularly reward players for doing work and give them a predictable way of seeing if they are doing too much or too little work. Obtaining ranks could also be a regular assignment to keep students playing and exploring topic space.

Chapter 5

Quests

Creating a sandbox world in topic space is a good step forward to allowing for exploration. However, this cannot be the end of things. Meandering through an open world can be entertaining but can eventually lead to exhaustion and kill any motivation to play. In addition, open worlds can be incredibly daunting; an infinite number of choices take an indefinite amount of time to decide between.[1] While ranks and skill levels help measure a players progress, they do not tell the player where to go.

The most common solution to this in open world games it the implementation of quests. Even when quests are simply things players can do on their own, it creates a goal and a sense of accomplishment when met. Quests also direct player effort into specific areas of the game.

Just as open world games need quests, courses need assignments. In a gamified course, there are a few extra benefits to using quests. Just as topics can be seen as sections of a chapter, stringing together topics is a chapter itself. Assignments

[1]http://en.wikipedia.org/wiki/Hick%27s_law

Quests			
Types	Lesson	Challenge	Project
Game Equiv			
Enemies	few mobs	miniboss	boss
Area	room / field	dungeon	final dungeon
Course Equiv.			
CS	tutorial / lab	small program	project
Writing	readings	response essay	literature review
Assesments	quiz	test	exam

Table 5.1: Basic Quest Types and Equivilents

can be seen as nothing more than quests the player must play for a grade instead of fun. While quests can be anything at all, there are a few types (5.1) that are most common. Using just these types, most traditional course work can be covered.

5.1 Lessons

- Most basic type of quest
  - in games, this would be just a single simple quest
  - go to location X, kill N of monster Y.
- In this course, lessons are the same as a nightly homework assignment
  - like an assigned reading or the math problems at the end of the chapter
- Lessons are a series of topics and pointgates

5.1.1 Structure

- lessons plow through

5.1.2 Point Gates

- Point gates act as the locked room between topics.
- skill points awarded totalled awarded per topic

5.1.3 Playing

Alternative Question Phase: As lessons are proposed, each group of tasks has the potential to be interuptted by a group of questions. This has the potential to interupt the workflow of a student. Instead, the questions can all be moved to the end of the lesson. Once a player runs out of tasks in a topic, the lesson stores the number of points they are missing for that topic and moves the player on to the next one. Then, once all the topics are completed, the lesson shuffles all the questions for all the topics. It then presents a questions the same way until the player has enough points to clear the point gate. In order to avoid excess work, the questions could be filtered so the questions from incomplete topics are asked.

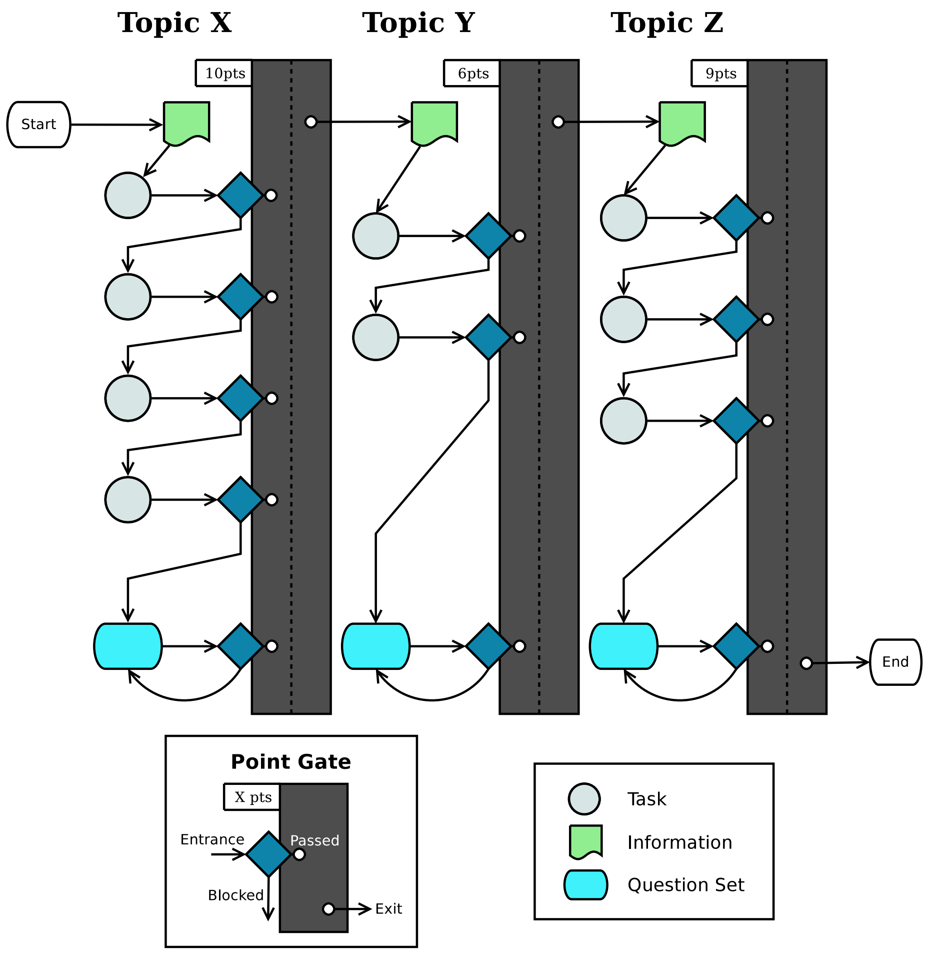

Figure 5.1: Lesson Flowchart

Example

Let's imagine someone is actually playing through the lesson laid out in figure 5.1. For the example, assume that every task is worth 5 pts and every hint is worth -1pts. You can follow along in figure 5.2.

The player starts by entering topic X. The first thing she sees is topic X's information, teaching her what she needs to know this topic. The player then sucessfully completes A1 awarding 5pts. However, this topic's point gate is 10pts so the she continues to the next task in topic X. While attempting A2, assume the player required 2 hints and only earned 3pts. This brings her total to 8pts, still not enough to pass through the gate. After completing A3 and awarding another 5pts, the gate opens and leads on to topic Y.

Assume that the player has already visited topic Y before. This could have been in a different lesson or through exploring topic space. Either way, when she was last in topic Y she completed B1 with 3 hints. Because of this, instead of attempting to solve B1 again, the player simply reviews the task, their hints, and their answer. The player can optionally see all the hints for this problem at no further cost. When she completed B1 the player was awarded 2pts. These 2pts count towards to point gate but are not awarded a second time.

Since she does not have enough points she moves on to B2. Once there, B2 gives the player a lot a lot of trouble. She starts by asking for some hints but still has no luck. In the end she requests and explanation for how to solve the task, earning only 1pts.

At this point the player has completed all the tasks for topic Y but has only earned 3pts of the required 6pts. In order to make up the remaining 3pts, she must answer questions about topic Y. For each correct answers the player earns an additional 1pts. Questions are shuffled and the player answers 2 of the 4 correct. This brings the total up to 5pts, still not enough to pass the gate. The incorrectly answered questions are shuffled and asked again. The player gets one more question right, reaches the required 6pts, and passes through the gate.

In topic Z the player is on a roll and she completes both C1 and C2 without any trouble. The player passes through the final gate and the lesson is over. In the end, the player gained 29pts, 4pts more than the minimum of 25pts.

5.2 Challenges

A challenge act as a slightly larger quest than the basic lesson.

Challenges provide very little in skill points but a lot of experience points. As a rule, player rank should steadily increase primarily from the completion of challenges more than individual lessons.

Unlike lessons, challenges provide no explicit hints.

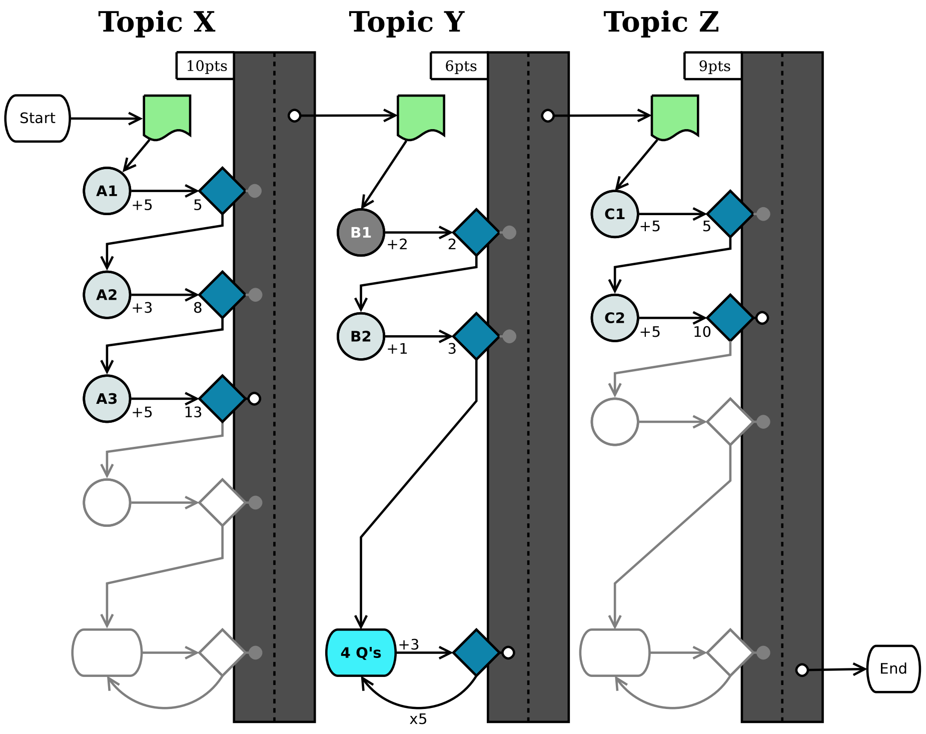

Figure 5.2: Example of a player moving through a lesson

Challenges

Challenges can also be used to group lessons and act as a quest chain for related lessons. The quest chains can be created by requiring certain lessons to be completed in order to finish portions or the entirety of a challenge. This creates in essence a tiny dungeon for a player to complete. Motivated by the challenge's goal, the player works through a series of lessons (rooms).

Alternatively, lessons can be treated as sidequests to challenges. As the challenge goes on, it may propose further reading or other things which can be done. These suggestions are not requirements towards completing the challenge. This could lead to lesson discovery, unlocking certain "challenge" or "bonus lessons" only when discovered inside a given challenge.

5.3 Projects

Projects give no hints on how they need to be completed. However, any knowledge needed to complete the most basic project should be covered in challenges. In this way

5.4 Quest Chains

Lessons as a rule are not included

The bulk of questlines are a group of challenges followed by a project. The challenges should be slowly teaching more advanced topics,

While not required, player. culminating with a project proving that the player can apply what they just learned.

There is no explicit requirement for the existence of questlines. They emerge from a combination of locks, visibility, and .

Chapter 6

Rewards

Part II

Implementation

Chapter 7

Requirements

7.1 Backend

7.2 Customization

7.3 Expansion

7.4 Recommendation

- this is one case where power is everything
- JVM based language could be the easiest way to go, but not java
- jruby or clojure could be the best bet
- python (not jython) could be used in a pinch but the way metaprogramming is handled in it as a language does not lend itself as simply

Chapter 8

Clients

8.1 Compatibility

*nix based systems offer the most flexibility if designing this for a CS course.

Custom clients/apps can certainly be written

- Web sites for the least whatever.
  - basic forms work for most lessons
  - more interactive pieces may need

8.2 Local vs MMO

Required internet connection

MMO style allows for students to play together

- sync
  - couchdb is a half decent option (any other server like it)

Chapter 9

API and Generic Data

- Decoupling topic space from the game world is vital
  - allows topics to be shared, mixed and matched, crowd sourced, etc
  - still allows custom ___
  - just like a map can be played on deathmatch or capture the flag

Chapter 10

Integration

While the major radical departure from a more traditional course is primarily in the handling of course work, this invariably affects every aspect of a more traditional course.

10.1 Lectures

In the case of an online course, lectures could easily be replaced by some sort of supplemental reading such as a textbook.

Potentially more interactive pieces.

10.2 Tests and Quizes

One of the main reasons to have tests and quizzes is to make sure that students are actually doing the reading and following along with the course. Tests can provide the instructor a snapshot of a student's current knowledge. However skill levels, ranks, and any other piece of information can be readily accessed and act as a snapshot instead.

That says, this framework does provide a way to create a test or quiz inside of topic space. By simply removing the hints and locking off a series of topics a region of topic space can be protected from wandering players. Since these topic regions are isolated from the rest of topic space, they can be thought of as a cave or dungeon in a traditional video game.By either using the normal lesson structure or creating a new one the player can play through these regions.

10.3 Enrichment

The entire purpose of using topic space is to encourage exploring concepts. While players can wander around on their own, there may be cases where instructors want to create enrichment lessons. Instructors can rapidly create new lessons available to everyone in the course or for a specific player. These enrichment lessons can be used to make up for missing work or just challenge a player who isn't being challenged enough.

Chapter 11

People

Students

- profile
  - basic information available, completely student editable
  - private, protected, public, and globally public
- the profile itself could potentially be carried from course to course
- course work can potentially affect aspects of the global profile, but very few
- taking a course creates a new section on the profile
  - private/protected
- profile tracks personal progress (private), peer progress (not as detailed, protected)
- acts as an informal internal resume

Course Masters (CM)

- The people actually in charge of the course
  - instructors
  - ta's
- also possess a profile though with no status tracking
- contacted for help with the course

- authorities (can update progress, award any given type of points, and create new supplemental assignments0
- instructor has additional root level control, can award it to course masters

Community

- Not entirely necessary though it can aide.
- a place for discussions and interactions outside of RL
- actively show who is working on what assignment and IM chat (opt in)
- place where assignments and what not are posted
- most generic questions should be posted (or mirrored) here
- answering another students question is good

Parties

Chapter 12

Tools

- shiny and pretty and intuitive
- progress bars
- make winners, onboarding
  - slow introduction to features
  - positive reinforcement
  - remove failing at first
- consistent terminology

Any skill can be hidden from a player. Visibility is not inherited from a parent skill. This means that which skills are immediately visible to the player can change while the mechanic remains the same. For example, the player starts with a list of skills they must accomplish while others remain hidden. The player works on the "while loops" and "for loops" skill. For a lower ranked player, it would make more sense to show each skill individually. However, as they progress these skills may be hidden while their parents "looping" and "basic coding" become visible instead.

A hidden skill does not mean the player cannot find this skill somewhere in their stats, it simply means that any skill points awarded will be shown going into the relevant visible skills and not everything.

12.1 Theme, Language, and Libraries

12.2 IDE

12.3 Other Tools

Chapter 13

Unaddressed

Part III

???

Chapter 14

Potential Problems

14.1 Content Creation

Assuming that a framework could be written to run this gamified course, the amount of effort it would take to create the content required to use this for for a single course is daunting. Each topic domain contains a significantly more excercises than are required for a given assignment. Then testing needs to be created to check the work submitted for each excercise. Add to that multiple levels of "hints" per excercise. Already writing a single question could be said to be a solid days work.

In addition, instructive text and questions most likely need to be created per topic. Challenges need to be written and dependencies need to be defined. Even if by chance it did not take an unresonable amount of time to create the content, there is still the issue of checking and testing all the created content.

14.2 Cheating

While not unique in having trouble with cheaters and plagerism, there is currently no counter in this course for either.

- encouraged group work makes it harder to check for uniqueness and even if two programs are similar it doesn't mean cheating
- exposed github repos means you can literally just copy
- people may get answers from previous years (higher cost of designing new problems)
- ideally though projects wind up being public assignments are not
- reuse

14.3 Play Testing

14.4 Domain Boundaries

The largest problem with a system like this is its at least partial reliance on computerized grading. Though a program can easily tell if something is exactly correct as planned, it is much harder to judge if a solution is correct enough.

14.5 Exploration versus habits

14.6 Falling Behind

14.7 Background

Unfortunately, as far as I can tell the current state of Gamification work and design does not lend itself to gamified coursework. It seems that most work done in gamification right now focuses primarily on the social and multiplayer aspects of gaming. The primary book I was intending on using as a source, Gamification by Design, is geared more towards adding game elements to existing products and marketing campaigns. Most gamification applications such as Foursquare and ChoreWars deal more with multiplayer design. Augmented Reality Gaming (ARGs) are also primarily focused on creating communities of cooperative players.

However, coursework is not a social activity. Even in the case of group work, the groups are atomic. It makes more sense to draw inspiration from single player and the edutainment genre than research and resources with a label of gamification.

In the end I drew from a variety of resources listed below.