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.

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

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.

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

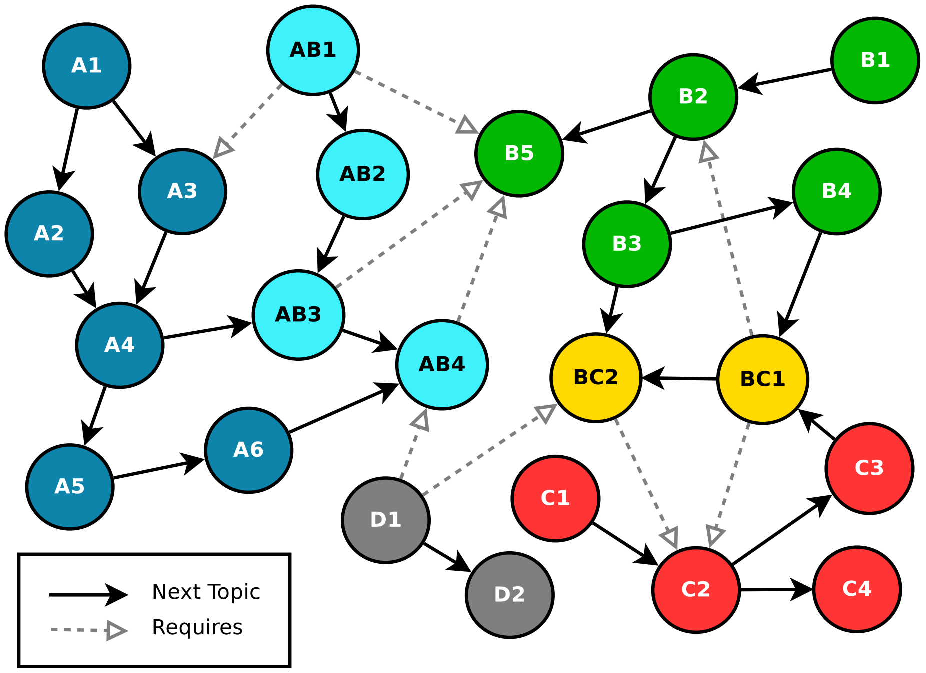

Figure 2.2: Topic Space

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