CS 120: Intro to Computer Programming II

In-Class Activity - 08 Top-Down Design - Day 3

Breaking Down Data

You can use the Top-Down Design strategy for your data - not just for your code!

To illustrate this, let's break down the data structures for a simple game. I suggest that we break down the original Super Mario Brothers (or choose a more recent, similar game if you prefer).

Let's see how much detail we can come up with about how the **data** of Super Mario is stored - I think you'll be amazed how much we have to keep track of!

As with all of the Top-Down Design ICAs, don't worry too much about how far your group gets. Turn in what you have, when the class is over. However, you might still want to **read** the rest of the ICA after you're done - to get more tricks about designing data!

Getting Started: The Grand Categories

As we begin to plan out our data, let's start with the largest, most general categories. I've broken the data into 3 categories to get you started; they should be enough to hold most of the data that Super Mario would need to keep track of.

- **Prototypes**

  These data describe how objects work in general. For instance, there might be a "goomba" object, which describes how Goombas move around, what their abilities are, and how they look. Another object might describe how Koopas work, or the Hammer Brothers, or the squid things in the water worlds.

  But it's not just enemies. We probably need data to tell us what Mario is capable of (and how he looks), as well as information about every single block in the world.

  However, these data objects do **not** tell us anything about a specific entity. So it speaks about a "brick block" - but not about where any bricks actually are.

- **Levels**

  Every level needs to have its design stored somewhere. This will have information about all of the static elements - the various blocks and such - but also information about where to spawn enemies, animated features, buttons, switches, links to other levels (including pipes within the world), etc.

  You can assume that this data is **full** of references to the Prototype objects.

- **Runtime Data**

  This contains everything we need to know about what's currently going on in the game. What level are we on? What is Mario's current position, speed, facing, and posture? What enemies are on the screen, and what are they doing? What objects are moving around? What blocks have been broken, or changed, or activated?

  Probably, this data has lots and lots of references to the level designs - and maybe to the prototypes as well.

There is an **almost endless** number of reasonable ways you might organize your data! There's nothing magical about these categories - but let's use them today, as our starting point.

(activity continues on the next page)

Activity 1

Starting with these categories as the grand overview, start breaking each category down into more detailed design. One of the simplest ways to do this is to give yourself a few examples to start with - and consider what you would need to store each of them.

But beware - your job today is **not** to build an encyclopedia of all of the Super Mario enemies! (If I wanted that, I'm sure that there's a Wiki I could read!) Instead, your job is to **design the data structures.** We are more interested in the **questions we might ask** about the enemies - not in the specific answers for any given enemy.

So, what are the things we need to know about the Goomba? Here's a short list to get you started, what else can you think of?

- Its graphics (all of its pictures, known as "sprites", and all of the animation sequences)
- What happens when it gets stepped on
- What happens if it is touched by a fireball
- What happens if it runs into another enemy of the same type

Notice that some of these sub-categories are still very complex! The graphics part, for instance, might have **many** smaller pieces inside it. That's not just OK - it is **great!** Organize your data into sub-objects, to make it easier to understand.

Make a list of as many fields as you can think of (remember, we want the **questions,** not the answers!), and turn that in.

Activity 2

So, you've figured out lots of questions about the Goomba. Now, let's see if you've overlooked anything. Think about several other enemies - for instance, the Koopas, the sun that chases you on desert worlds, Piranha Plants, or even Bowser himself.

Almost certainly, you've overlooked some important questions. Update your question list, until you think that you can store all of the data that's important about all of these types of enemies.

Turn in the updated list.

Activity 3

This is Top-Down Design - let's break something complex into something a little simpler! Take one of the data fields that you've already discussed, and break it into sub-categories. For instance, you might break down the Graphics question I suggested. Or mabye you have something else you'd like to break down?

Do a couple of rounds, breaking the data down into simpler pieces, and then breaking some of it down again, into something even simpler.

Turn in the more detailed design you've come up with.

Activity 4

We still haven't finished thinking about Prototypes! Let's spend a little time thinking about bricks.

In theory, you could make bricks and enemies be the same thing - but really, it seems to me that they are **very different.** Probably, you want to start an entirely new class, to store the Prototype information about blocks that make up the world.

Start thinking about this new class: what types of blocks will make up the world, and what are the fields you need to store (that is, the "questions you need to ask") about each block?

Come up with a list of fields, and turn that in.

Activity 5

Often, two object types - even if very different - have a few things in common. For instance, enemies and blocks on the map both have graphics! In this case, the typical solution is to define a smaller class - which holds only the things that are shared.

In later classes, you'll learn about "child classes," which allow a class to inherit some implementation from another, simpler class - but for now, let's just use references. Each "enemy" object will have a reference to a "graphics" object - and the same will be true of blocks.

Think about what fields you have, that might be in common between enemies and blocks. Update your design for both object types, and turn those in.

Activity 6

At last, we're moving past the Prototype data! Start sketching out how you would store the data for a **Level.**

Remember: this must not store anything about what's currently going on in the game. So I don't want to know anything about where each enemy is **right now** - although I probably want to know each enemy's **starting point** on the level.

There are lots of things to think about, in a level design. Not only where the blocks are, but also how the level links to others. Where are the pipes, and where do they go? If a pipe takes you somewhere, is that another level entirely, or should you map that as simply jumping you to another location in the map? Are some blocks invisible? Etc.

As much as you have time, keep exppanding your data design - into the levels, or even, if you wish, into the actual runtime data of your program.

Have fun!