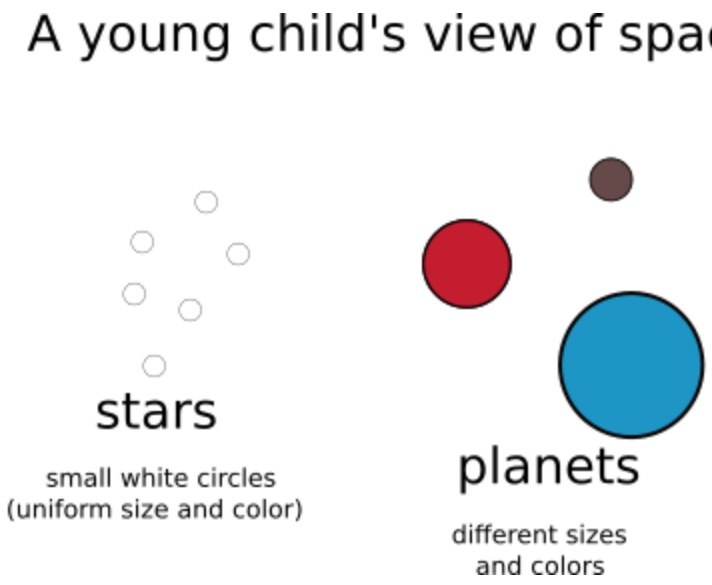
Stage 2: video 3

Scene 1: On-Set

To demonstrate the power and flexibility of the alloc-init pattern, we'll look at a more complex example.

We'll design our init methods to simplify the process of generating objects for a specific application. As with any system of initializers, the goal is to be able to create objects, with the required configuration, as simply, *clearly* and easily as possible.

<MOTION>



Our example is a model of how a child might see outer space.

The most common objects are stars, which are a uniform size and always white.

Planets are less common and come in a variety of sizes and colors. The code that will be populating “space” will need to generate a large number of star objects and a lesser number of planet objects.

</MOTION>

Scene 2: Screencast

For this “Stars & Planets” example, we’ll be expanding on our “Circle” class from a previous video.
The project will be getting larger so, to keep things manageable, we’ve broken Circle out into its own class modules. Other than that, Circle is how we left it previously.

So, on to modeling “space” objects. We have a diameter property in Circle which can serve as “size”, but we need a “color” attribute in order to generate “stars” and “planets”

We’ll create a subclass of “Circle”, called “ColoredCircle” and add a “color” property to the subclass.

ColoredCircle’s color property will be an NSString since we’re programming Foundation on the at the moment. A color used in a view on iOS be a UIColor, on

the Mac it would be an NSColor, though the interfaces for UIColor and NSColor are nearly identical.

After adding the color property, we declare the designated initializer for ColoredCircle. This method takes a diameter and color parameter and calls Circles' designated initializer with the diameter parameter to complete the instance initialization.

As in Circle, we'll override *init* so if a ColoredCircle is created without specifying its properties, it will be initialized with default values. The default colored circle will be an "earth" with a diameter of 6 and color "blue"

We choose 1.0 as the size for all "stars". So stars will be white ColoredCircle objects with a diameter of 1.0.

Populating "space" will require creating lots of stars, so we'll simplify the job by adding a factory method. "+star" will generate a "star" without the need to specify size and color.

The "+star" factory class method implementation is simply a wrapper for the designated initializer called with values appropriate for a star.

For anyone reading the code, the call: "[ColoredCircle star]" will more clearly communicate that a star is being generated than [[ColorCircle alloc] initWithDiameter:1.0 andColor:@"white"]

Finally, we'll add a call to +star in main, and step through the process of object creation with the debugger.

We'll add a breakpoint at the first call to +star in main() and follow the process of "star" creation using LLDB.

We'll be using the LLDB command-line for this example:

1. "step" or "s" is used to follow execution into method or function calls (the same as the step-in button)
2. "Stepping" bring us to the implementation of +star
3. Using "step" again brings us into the designated initializer for ColorCircle
4. "s" again brings to Color's designated initializer
5. From here we'll use the "next" command (equivalent to the the step button) to follow execution through all the initialization and back to +star
6. Examining our variable "s" using "p s", we see that it's a ColoredCircle object and we're given it's memory location.
7. Using "p s.color" and "p.diameter" we can examine the properties individually.

The next step would be adding factory methods for other space objects, but we'll leave that for a code challenge.

Code challenge

1. We need a ColoredCircle factory method for "asteroid" which has color "grey" and a diameter of 1.5: write the declaration for this class method.

Ans:

```
+(ColoredCircle*) asteroid;
```

2. By calling the designated initializer, code the implementation for this class method.

Ans:

```
+(ColoredCircle*) asteroid {
return [[ColoredCircle alloc] initWithDiameter:1.5 andColor:@"grey"];
}
```