Week 2

Content:

- Rapid Prototyping
- Video: Designing Kodu Games
- Brainstorming
- Identifying Risks
- The 400 Project
- Design Patterns in Games
- Kodu Game Lab Recipes
- Week 2 Task

Rapid Prototyping

The more formal approach to game design

Iterative process

Last week we looked at the informal approach to game design. This week, we take a look at a game development approach that's more formal, with clearly outlined steps designed to encourage you to consider your game design more broadly.

An iterative process, is a commonly adopted cyclical method you can use to approach designing a game.

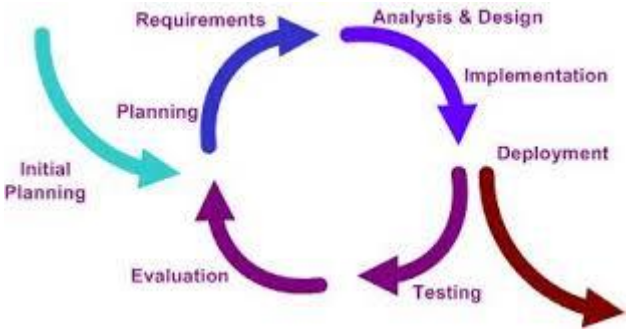


Image from Wikipedia – a typical Iterative Development Model

1. The Planning and Requirements steps are a mission statement of sorts. You'll need to establish the known goals for your game, what you'll need to build it.
2. Next, you design your game, perhaps drawing a paper prototype of it before you start building.
3. Once built, you test the game.

Seems a little like the informal design we explored last week doesn't it?

Well yes, it is like the informal process, except for a few key factors.

1. Firstly, you establish clearly up front what your goals are, and then with each round of testing and evaluation, you return and revisit those goals. There's still a healthy amount of trial and error involved, but this method allows for more formal reflection.
2. As each cycle is completed, you further refine your statements established in the first step. This is often necessary as your hopes for how the game will function, and the reality of how the game actually plays, are often a little different. You may find that as you progress, some of your initial aspirations for the game are either unnecessary, unachievable, or require a number of improvements.
3. With a more formal approach, your known goals will always been front of mind as you progress with development. Not only will you be considering whether they are being met with each new cycle, you'll be determining your games re-playability each time you begin testing.

How to Prototype a Game in Under 7 Days by Kyle Gray, Kyle Gabler, Shalin Shodhan, Matt Kucic

Whether you're taking a formal or informal approach to design, building something quickly introduces challenges that will be present with either method. If you have a time limit ahead of you, like our challenge last week, there are a number of factors you'll want to consider:

a. Choose a theme that underlies your game. It may be for example, that a version of the universally known schoolyard game of tag is the underlying structure, predator and prey. Or, you might decide that the game centers on audible triggers. A theme will help give your game the underlying structure it needs to be engaging, and you'll save time during the games development if you identify it early on.
b. Don't dwell too long on an idea that's not working out – try something new, start something different and come back to your original idea later.
c. Just because you want to build something quickly, doesn't mean you can skip important steps. If your game hasn't been well thought out, if the actual experience of playing the game is less than engaging, no amount of mood lighting, music or clever use of colors will hide that fact.
d. Just because you want to build something quickly doesn't mean you should ignore the mood lighting, music or clever use of colors! A game that looks and feels polished will be more absorbing. It won't save a badly designed game, but it can greatly enhance a well-considered one.
e. Embrace failure! We learn from our mistakes, it's as simple as that. The more you embrace your failures as golden opportunities for learning, the faster you'll develop as a game designer.
f. Never underestimate the power of the simple game. Your game doesn't have to have a thousand different variables to be fun. Pac Man and Pong are great examples; despite having been around for years now, new versions of these games keep emerging because they are simply engaging … or is that engagingly simple? Simple games can be completely and utterly absorbing, it doesn't have to be complicated to be fun.
g. Having an end goal will not only make your game more engaging, you'll find it easier to work toward a target. I've noted a number of RPG attempts in particular built with Kodu that have suffered from this problem. You wander here, you wander there, with no clear idea of what you should be doing. Sure the terrain and mood lighting are impressive, but what am I doing here? People get bored fast, so make sure the goals are clear up front.

Video: Designing Kodu Games

In part 2 of our interview with Charles Howell, we discuss the process of making great games with games, specifically the process that Charles himself uses.

Brainstorming

You can't schedule creativity, there, I've said it. You can schedule the work that goes into building an idea into a solid workable prototype, you can schedule the different aspects of the development cycle you intend to follow … but coming up with new ideas?

You might find these are best uncovered in the traffic on your way to work, while your mind wanders while doing the dishes, or reading a book before bed. Personally, I find with Kodu Game Lab, it's when I'm pulling apart somebody else's game that I'm most often inspired by a new idea.

And where does brainstorming fit in? Well, there are a number of ways you can try to kick start the creative process. It might be simply by messing around with Kodu, or you might find making a list of the

things you want to achieve in your game brings out some ideas. There are numerous approaches to brainstorming, the trick is finding the one that works best for you.

Here are a few suggestions to get you started:

Automatic writing

Simply start writing as quickly as you can, all the things you want in a game. Don't stop for spelling or punctuation, don't stop to consider or flesh out your ideas. Just write as fast as you can, everything that pops into your head on the subject. Set yourself a time limit, a couple of minutes perhaps. Now, stop writing, sit back, and put a circle around anything that catches your eye.

Switch off

Sometimes just switching off for a while is the best way for new ideas to spring forth. If you find the new ideas just aren't coming, take a break and go for a run, have a bath, and give your mind a rest for a while. You'll be surprised how often getting off the task for a short time can be the best way to inspire a new idea!

Messing about

Playing with Kodu with no clear goal is another way you can find inspiration for a new game. Don't head in with any clear intentions in mind. Simply start up Kodu, and start building! Try things you'd normally not consider just to see what happens. You can discover a lot about Kodu when you're not placing yourself under any pressure – and you might just find a new idea will surface from all your unplanned messing about.

Paper prototype

Sometimes using a different medium can a great way of inspiring new ideas. I know for example, that I approach problem solving differently when I'm typing, as opposed to when I choose to sit down with paper and pencil. As I tend to type very fast, using paper and pencil forces me to slow down and more closely consider the language I'm using and the ideas I'm exploring. The same applies to designs on paper with Kodu. Draw an approximation of your world and then start putting in characters and ideas. Once you're done, try recreating it in Kodu.

Identifying Risks

When attempting to create any game it is important to consider the design risks and this is especially important when creating games with Kodu Game Lab? Is this game technically possible to make with Kodu Game Lab?

Is there any aspect of the planned game that is not possible to make with Kodu Game Lab?

Can the resources required be kept to size and level of complexity to make it playable with Kodu?

Can the object or bot be programmed to act in a required way?

Can a world be created to in the required form?

When generating ideas for your games it is useful not to be concerned with reality but at some stage it needs to determine if it is actually possible to make the desired game with Kodu. Obviously, for some game ideas that are variations of other Kodu it will be relatively safe to assume that the game will be possible to make. Other ideas, say, creating Second Life with Kodu, will obviously be impossible. So what can we do to minimize design and technical risks?

The Process

1. Identity the major technical and design risks of your design

Once you have decided on the game that you plan to create is important to identify the parts of the game that might not be possible.

2. Build a prototype of the risks

Start by building the parts (the risks) of the game that you feel might not be possible. It is not necessary to build a complete game at this stage or get this part of the game exactly right. This stage is all about determining if this risk is real or not. With Kodu it is not only necessary to determine if it actually possible but also necessary to determine how many resources this part of the game uses.

3. Test

Play testing is the most important part. Testing at this stage only tests the specific identified risks.

4. Modify (or abandon) as needed

Once you have evaluated the risks and determined what is possible, you can either proceed as planned, modify your design with regard to the discovered limitations or abandon the project.

Spending time identifying the risks in your game design will save both time and frustration when making larger and more complex games with Kodu Game Lab.

The 400 Project

The 400 Project is attempting to find and list 400 (just a rough number) rules that apply to Game Design. The project began in 2003 and so far has compiled more than 100 rules. The project is managed by Hal Barwood and Noah Falstein. Most of the rules have been written by Noah Falstein, although anyone can submit a rule.

Each rule has a name, a statement (description) and a domain to which the rule belongs. Each rule has an example or two from well-known games as well as counter-examples that show the consequence of not following the rule.

For example:

Provide Clear Short-Term Goals

Always make it clear to the player what their short-term objectives are. This can be done explicitly by telling them directly, or implicitly by leading them towards those goals through environmental cues. This avoids the frustration of uncertainty and gives player's confidence that they are making forward progress.

Noah suggests that rules are useful when you are stuck with your game design or evaluating the game you are making against each rule in the list.

Good Game, a TV program in Australia, ran a short story on the 400 Project, last year but the clip is no longer available on YouTube.

During the coming weeks we'll look at some of these rules and how they apply to the game experience. We would love to hear your thoughts on these rules.

Design Patterns in Games

Patterns are used by designers to solve problems using specific known solutions and that have been previously proven to work. In 1977, Christopher Alexander wrote the book "A Pattern Language" that outlined how patterns could be used in architecture to enable anyone to design buildings. Design patterns were popularized by the Gang of Four for use to solve common object oriented programming problems.

Design patterns:

- give a general solution to common problems
- are universal to any programming language/environment / framework
- outline the reasons why the pattern is appropriate
- create a shared language and understanding

Staffan Björk & Jussi Holopainen have detailed about 300 game design patterns on their website and in their book. The website only has a basic description, an example and the relations for each pattern but it is still very useful.

Example from the website: Boss Monsters

A more powerful enemy the players have to overcome to reach certain goals in the game.

Sometimes defeating the Boss Monster can be a goal in itself, but usually Boss Monsters are used as sub goals in the game and the high-level goal is of another type of goal. Boss Monsters are almost always used to structure the progress of the game.

Game Design Patterns can positively influence the design choices of game designers by enabling the games to have "authentic game experiences". In particularly investigating the patterns for narrative structures, predictability and immersion, such as tension or planned character development, may be useful in generating ideas for your games.

Many of the patterns and descriptions may seem obvious and some of the patterns may not be applicable to Kodu Game Lab however a deeper knowledge of the design patterns in games will undoubtedly help you create better games.

In weeks 3, 4 and 5 of this course we will unpack some of these design patterns in greater detail looking at how they can be used to build the game experience.

Kodu Game Lab Recipes

(and other ways to make life easier)

While Kodu Game Lab's visual programming language is rather intuitive there are a number of ways that you can minimize the learning curve.

1. Modify or build upon existing games.
The simplest way to create a game with Kodu Game Lab is to modify or improve an existing game. Any Kodu game played can be edited and modified or improved upon make it better.

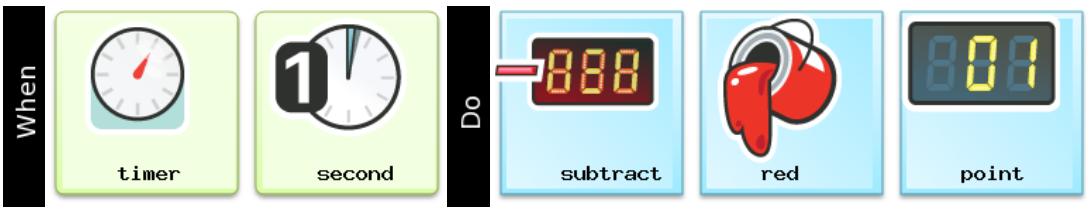
2. You can look at the code of any game and see how it was done.
Alternatively you can build your own game from scratch but learn from others by examining their code. Sometimes it may be obvious where the code from the game is located and

3. Copy and paste code from one project to another.
Kodu allows you to copy and paste objects and bots from one project to another. This enables you to mix and match from different games to create your new game. You can also use the clear tool to remove bots and objects from an existing world that you plan to use as a basis for a new game.

4. Use programming recipes to implement specific elements.
Programming recipes are code snippets that perform a certain task. Recipes are usually short language specific code that achieve a specific task. Often programmers would intuitively know the code to achieve these relatively simple tasks however when starting out, are having understood the basics of the language, recipes are an easy to build your knowledge of syntax and common techniques.

For example, a recipe to add a countdown clock to your game.

To create a countdown timer for 10 seconds use the following Kode to an object that won't be destroyed during the game! Also set the red score to Quiet.



In most common languages cook books of language specific recipes are available on websites and in book form.

Week 2 Task

This week we're offering the choice of two tasks depending upon the time you have.

A. Improve an existing game

Find an existing game, whether it be a game you have previously made, a default game that is bundled with Kodu Game Lab or a game you've found on the community website and then improve it. This may include adding new bots to the game, expanding its scope or improving the AI and game play, it's up to you.

OR

B. A more complex game

This week, with a more formal strategy in game design in mind, we're asking you to take the next step in the development cycle. Using the steps outlined this week, we'd like you to write a list of achievable requirements for your game first, perhaps a mission statement, make a few notes about the design, and then follow the iterative process outlined in this week's course guide.

There are no time limits on this week's challenge. You simply need to submit your game, finished or unfinished, with your notes. It doesn't matter whether your notes are a few simple bullet points, or a considered paragraph or two. What's important is that you have reflected on the development of your game more strategically.

Oh, and it's worth noting that we're particularly interested in your failures! That's right, if you've found little success on your initial attempt, we still want to hear from you, so we can all learn from each other's mistakes!