Hacking Education with Virtual Microworlds

Pascal Chatterjee

May 30, 2014

Abstract

Contrary to popular belief, the scientists working in their laboratories are not the members of society who learn the most in their daily lives. The members of society who do *by far* the most learning in their daily lives are children.

Children tend to walk by the age of 18 months; they can express themselves in the extremely complex system known as natural language by the age of two years; and they have developed a theory of mind after spending a mere four years in the world. And they manage all of this before entering formal education.

The progress of even the hardest working university student pales in comparison. This is not just due to the difference in age between adults and children. It also has to do with the environment in which we learn.

Microworlds are environments in which adult minds can **construct** knowledge in a similar way to children. This paper explains the ideas behind microworlds, describes two implementations of them, and discusses whether they worked and if they will ever be a feature of mainstream curricula.

The microworld created for this paper, *Narrative Roulette*, is both an engaging and effective learning environment for teenage students. But it won't be until teachers have the technical skills to deeply understand virtual microworlds, and authorities believe in constructionist learning, that microworlds will be adopted by the mainstream.

Contents

Chapter 1

Ideas

1.1 What is a microworld?

Imagine this. You're hungry. It's late in the day and you need to decide what you're doing for dinner. It's a familiar problem.

What does the solution space for this problem look like? The usual way to figure this out is by examining a single solution and then seeing how its properties vary.

A single solution to the whats-for-dinner problem is a single meal. So the solution space encompasses everything that can be considered a single meal: everything from a steak to a snack bar. And of course there's the null solution: not to eat anything at all.

Ideas: open-ended goals

We explore this solution space every time we plan a meal. We work under certain constraints, such as availability, our personal preferences, diet plans and even the cultural acceptance of the meal in question. The set of things we are likely to eat is far smaller than the set of all things that we *could* eat.

That said, we can also be creative with our solutions. Though our goals must conform to a certain shape (say roughly 1000 calories of nutrition), they are also **open-ended**. There are an infinite number of ways to get 1000 calories of nutrition, even taking into account all of the constraints we just mentioned.

Of course, we rarely think about all of these possibilites. We tend to stick to what we know, reducing an infinite vastness to the comfortably familiar.

I cook pasta dishes a *lot*, and I doubt I'm the only one.

If I decide to combine pasta with meat and pesto, I have come up with a potential solution to the whats-for-dinner problem. This is a creative idea in the domain.

To implement my creative idea, I have to cook.

Implementation

What kind of meat shall I use? Shall I make pesto myself or buy it in a jar? What kind of pasta shall I use, and how should I prepare it?

When I make these decisions, I realise my abstract idea with a concrete implementation. I turn the creative idea in my head into something I can put on my plate.

In this case, let's imagine that the idea of pasta is reified into boiled, fusili pasta; meat is reified as fried bacon; and pesto is reified as the home-made kind.

Evaluation

Once my reified idea is on my plate, the evaluation phase begins. If my solution contains roughly 1000 calories, we can call it a meal. But is it a satisfactory one?

Like the solution space, the evaluation space is infinite. I may have achieved my nutritional goals, but failed my goals of taste. Or maybe my meal was both nutritious and tasty, but it just wasn't *novel* enough.

In the end, my evaluation of my meal is evident in what I do the next time I try to solve the whats-for-dinner problem. If I prepare the pasta and the bacon in the same way, but change how I make the pesto, then evidently my pasta and bacon implementations were good enough, but I felt my pesto implementation could be improved.

Learning is a process

By **implementing** my **creative idea** as a consumable "product", and then **evaluating** it by consuming it, I have *learnt* about what works and what doesn't, in the whats-for-dinner domain. I have learnt that I can make tasty pasta and bacon. I have learnt that my pesto needs improvement, and my first attempt should have given me some hints about what I should try next time.

> Learning is what happens when someone executes an **idea-implementation-evaluation loop**, and uses outcomes to guide future iterations.

> **A microworld is an environment that enables a person to iterate an idea-implementation-evaluation loop within a certain domain.**

A supermarket, kitchen and dinner table is a microworld for learning how to cook meals, for learning to solve the whats-for-dinner problem.

1.2 What kinds of microworld exist today?

By the age of 21, the average young person will have spent as much time playing video games as they will have spent in a classroom[1]. A young person's presence in the classroom is required by law. Their presence in front of the screen is entirely voluntary.

What is it that allows games to gain attention in such huge amounts, *without* the threat of punishment if young people don't comply?

What is a game?

Let's find out what a game actually is.

According to Jane McGonigal, a game designer and writer, the four defining traits of a game are the following:

1. "The **goal** is the specific outcome that players will work to achieve. It focuses their attention and continually orients their participation throughout the game. The goal provides players with a sense of purpose."[2]
2. "The **rules** place limitations on how players can achieve the goal. By removing or limiting the obvious ways of getting to the goal, the rules push players to explore previously uncharted possibility spaces. They unleash creativity and foster strategic thinking."[2]
3. "The **feedback system** tells players how close they are to achieving the goal. It can take the form of points, levels, a score, or a progress bar. Or, in its most basic form, the feedback system can be as simple as the players' knowledge of an objective outcome: "The game is over when..." Real-time feedback serves as a promise to the players that the goal is definitely achievable, and it provides motivation to keep playing."[2]
4. "Finally, **voluntary participation** requires that everyone who is playing the game knowingly and willingly accepts the goal, the rules, and the feedback. Knowingness establishes common ground for multiple people to play together. And the freedom to enter or leave a game at will ensures that intentionally stressful and challenging work is experienced as *safe* and *pleasurable* activity."[2]

To summarise, a game comprises of the **voluntary** attempt to achieve a certain **goal**, according to a set of **rules**, with **feedback** on how close you are to that goal.

Are games microworlds? Or are microworlds games?

Let's compare this with our definition of a microworld, from the previous section. A microworld is an environment that supports a loop comprising of:

1. An **idea** that a student comes up with *themselves*.
2. An **implementation** of that idea, according to the *rules* of the microworld.
3. An **evaluation** of the implementation, that gives the student *feedback* about the quality of their implementation and idea.
4. The evaluation generates further ideas, and the loop continues.

As we can see, there seems to be an correspondence between microworlds and games. An **idea** in a microworld corresponds to a **goal** that a student chooses **voluntarily**. The **implementation** of that idea must conform to certain **rules**, set by the *structure* of the microworld. And the purpose of **evaluating** a student's implementation is to give them **feedback** about it, so they can improve their future ideas.

But the two are not exactly equivalent, as a microworld's **idea** conflates the **voluntary** and **goal** traits of a game. Every microworld is a game (each microworld contains all four traits of a game), but not all games are microworlds.

> Only those games that allow the voluntary choice of goals can be considered microworlds.

Flappy Bird vs Minecraft

Wikipedia says this about the gaming phenomenon known as Flappy Bird:

> "Flappy Bird is a side-scrolling mobile game featuring 2D retro style graphics. The objective is to direct a flying bird, which moves continuously to the right, between each oncoming set of pipes without colliding with them, which otherwise ends the game. The bird briefly flaps upward each time the player taps the screen. If the screen is not tapped, the bird falls due to gravity. The player is scored on the number of pipe sets the bird successfully passes through, with medals awarded for the score."[3]

Though players play Flappy Bird **voluntarily**, they have no say in their **goal**. They just have to keep flapping, or they die and the game is over. The simplicity of Flappy Bird's **rules** (just flap) and the sophistication of its **feedback system** (if I'd flapped *slightly* earlier, I'd be alive!) are what make the game addictive[4].

This means that Flappy Bird qualifies as a game, but not as a microworld.

The videogame called Minecraft, on the other hand, is defined by Wikipedia as:

> "Minecraft allow[s] players to build constructions out of textured cubes in a 3D procedurally generated world. Other activities in the game include exploration, gathering resources, crafting, and combat. Gameplay in its commercial release has two principal modes: survival, which requires players to acquire resources and maintain their health and hunger; and creative, where players have an unlimited supply of resources, the ability to fly, and no health or hunger."[5]

The goal of Minecraft's **survival** mode is, unsurprisingly, to survive. This goal is non-negotiable. However, Minecraft also has another mode, in which players have neither health nor hunger. This means that there is only one reason for playing this mode of the game: the joy of building things from virtual, textured cubes.

In this particular case, of a specific mode of a specific game, we have an environment in which players can *choose their goals with total freedom.* This makes Minecraft's **creative** mode a model example of something that is both a game and a microworld.

What other activities can be microworlds?

We've established that microworlds are a subset of games - those activities that contain all four of McGonigal's traits listed above. Of course, the set of all games is wider than just video games. Let's look at some other activities that could be described as microworlds.

Jamming

Consider a few people gathering in a room, each with their own musical instrument, jamming together. There is a **goal**: to make music that sounds good. There are **rules** (or *constraints*): of timing (4 beat bars), chord sequences that sound good together, the traditions of the genre, the expectations of the potential audience, etc. **Feedback** is instant: the musicians can tell if something sounds good, *while they're playing it.* And of course, jamming is **voluntary** in the vast majority of cases.

This means that jamming is actually a game, even though we wouldn't usually describe it that way. Is jamming also a microworld?

For a game to also be a microworld, it has to allow voluntary choice of **goals**. Though the overarching goal of jamming is to make good music, the group can work towards that with subgoals.

Perhaps the group decide the first step is to come up with a catchy chorus. The guitarist might have some ideas for chords she wants to try; the singer has some idea of what words he wants to sing; the drummer has some ideas about a beat, and so on.

These **ideas** are **implemented** immediately, and **evaluated** soon after, by the musicians who judge whether what they're doing is working. If it isn't, they update their **ideas**: the drummer tries a different beat, or the guitarist switches chords, and they iterate. Otherwise, they move on to another part of the song.

In this light, jamming can be described as a microworld, as it shares its basic structure with something like Minecraft, even though, on the surface, the two activities look very different.

Creative writing

Writing fiction is a microworld too. A lone writer has an **idea** in her head for a story. It probably isn't the whole story, word for word, fully-formed in her head. It's probably a rough idea, like: "dinosaurs on the loose in zoo, people try to escape" or "criminal couple rob banks and are then shot by the police".

The smaller ideas that make up this grand vision might be: "the people escaping the dinosaurs are archaeologists", or "wouldn't it be cool to have a scene where a rampaging tyrannosaurus rex destroys a skeleton of its own species?".

The **implementation** of these ideas takes the macro form of scenes, characters and events; and the micro form of the words used to describe them. **Evaluation** occurs when the writer reads back what she's written, a process that the author David Foster Wallace once called "feeding the wastebasket"[6].

Again, this makes writing a game, and the free choice of goals, or ideas, makes it a microworld too.

Programming

The microworld of programming is the one I have most experience with. An **idea** in this world could be something like: "I want to build a system where users can vote for things". Smaller ideas could be: "the system should work on mobile phones" and "the page should update by itself".

An **implementation** could use a HTML5 front-end, a Python back-end, and HTTP and WebSockets to communicate between them.

The **evaluation** would be testing the system, first in unit and end-to-end automated tests, and finally by real users in a production environment.

Many programming languages support a REPL environment, which stands for **R**ead **E**val **P**rint **L**oop. In a REPL, users can try out **implementations** of their **ideas**, in code, and have them **evaluated** immediately. This provides an additional microworld at the micro-level, embedded within the macro-level microworld of software engineering.

For example, when working in Python:

```
[1] >>> ",".join([d for d in range(10) if d % 3 == 0])
TypeError: sequence item 0: expected string, int found

[2] >>> ",".join([str(d) for d in range(10) if d % 3 == 0])
'0,3,6,9'
```

When we **evaluate** the **implementation** at line `[1]`, the Python REPL throws a `TypeError`, saying that it found an integer where it expected a string.

This gives us feedback to update our **idea**, which leads to the different **implementation** at line `[2]`. When this is **evaluated**, we get back `'0,3,6,9'` which is ostensibly what we wanted, and we learned something about the `string.join` method in the process.

Degrees of Microworld

So we have a bunch of environments that we think are microworlds: sandbox video games (like Minecraft), jamming, creative writing and programming. They all seem to be games (or at least have the potential to be), and as ideas can be chosen freely, we argue that they are microworlds too.

But how "free" are they, and if we gradually restrict this freedom, will we eventually end up with things that are no longer microworlds?

Let's take our programming example. Imagine the project is a commercial one, paid for by a client with certain requirements. These requirements include:

- The front-end must look exactly like the given mockups.
- The front-end must be an iOS app.
- The back-end must be written in Java and use servlets.
- The entire project must be production ready within a week.

Do we still have a microworld? The person or team working on the project is severely restricted in what **ideas** they can have. They have to conform to the mocked-up design, write it in the specified front and back-end languages, and can only do things that will fit in the given time-frame.

As a result, the solution space is extremely narrow; there's probably only a few ways to do anything, and maybe only one that could work given all the constraints. So the team only really **implement**, and the client is the one who will **evaluate**.

Just implementing does not a microworld make. It doesn't even qualify as a game, as participation is not **voluntary**, and the **feedback system** is probably far from real-time.

> The more freedom allowed in an activity, the more of a microworld it is.

Ideally, the only constraints are those imposed by the structure of the microworld: in programming we are limited by our technology; in writing or music we are limited by our own abilities and the culture in which we live.

But an activity can still be a microworld, even if we have a deadline and have some constraints over those imposed by the environment. Writing a story can still be a microworld even if it has to be done by the end of the month, and must include dinosaurs and a happy ending.

Because the solution space is still wide-open, **ideas** can still be had. But once scenes, characters and actions start being heavily constrained, the potential for **ideas** will reduce, and the microworld will become just another job.

1.3 Microworlds enable learning

John Dewey

112 years ago, John Dewey, a philosopher of education, wrote:

> [...] the school in turn will be a laboratory in which the student of education sees theories and ideas demonstrated, tested, criticized, enforced, and the evolution of new truths. {dewey:p55}

Dewey conceived of a school consisting of laboratories and studios, where students would be able to construct things and experiment with their own hands, instead of constantly being told to memorise what to do and how to do it.

He wanted to see **ideas** demonstrated (or **implemented**); tested and criticized (or **evaluated**); for the evolution of new truths (or the updating of ideas through iteration).

Jean Piaget

According to Edith Ackermann's analysis of the work of Jean Piaget:

> To Piaget, knowledge is not information to be delivered at one end, and encoded, memorized, retrieved, and applied at the other end. Instead, knowledge is experience that is acquired through interaction with the world, people and things. {ackermann}

This is because, to Piaget (according to Ackermann):

> Kids don't just take in what's being said. Instead, they interpret what they hear in the light of their own knowledge and experience. {ackermann}

Imagine an adult and child standing near a fire. According to Ackermann's analysis of Piaget, Piaget would say that the child will not refrain from touching the flame just because the adult tells the child it will burn their hand.

Once the adult has gone, the child will reach for the flame regardless. They will only abort their attempt to touch the fire once its heat on their hand becomes uncomfortable.

Here is our first hint at *why* microworlds could be useful for learning. It's because microworlds are environments in which rich experiences can be had (experiences such as those described by Dewey). Microworlds are also safe places to have those experiences - hurting yourself in a game is far less painful than doing so in real life.

Seymour Papert

> Constructionism means "Giving children good things to do so that they can learn by doing much better than they could before." [...] Instructionism is the theory that says, "To get better education, we must improve instruction."{convsinst}

> Constructionism [...] shares constructivism's connotation of learning as "building knowledge structures" irrespective of the circumstances of the learning. It then adds the idea that this happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity, whether it's a sand castle on the beach or a theory of the universe. {sitconst}

Piaget thought that knowledge is **constructed** by the learner. We build **knowledge structures** in our minds, from concrete experiences - interactions with the world, people and things.

Papert added to this theory, to come up with what he calls "constructivism" - the idea that knowledge structures are best constructed when the learner is building a *public* entity.

I believe the inclusion of the word "public" is important. I think it has to do with the **evaluation** stage of our microworld loop.

If an entity is public, it will be evaluated in public, by more people than its creator. This allows for richer feedback than if the creator was the sole person in charge of evaluating their work.

In practice, this means that products of microworlds (whether they are songs, stories or programs) should be shared with at least one other person than the creator, to be considered valuable.

> Now one can make two kinds of scientific claim for constructionism. The weak claim is that it suits some people better than other modes of learning currently being used. The strong claim is that it is better for everyone than the prevalent "instructionist" modes practiced in schools. A variant of the strong claim is that this is the only framework that has been proposed that allows the full range of intellectual styles and preferences to each find a point of equilibrium.

Microworlds allow students to construct knowledge *in their own way.* They offer building blocks that each student can use in their own personal style. And the best way to showcase that style is in public.

According to Piaget, children learn by interpreting their experience in their own way, by updating their own internal knowledge about a subject in a *constructive* way.

According to Papert, the best way to be construct knowledge is in the process of constructing a public entity. This means that, learning to draw is best achieved

by actually drawing, according to constructivism, but to satisfy constructionism, those drawings should be shown to others.

When the drawings are shown, the creator gets feedback. They construct the knowledge of how their ideas, mediated through their implementation, is evaluated by other minds. They use this constructed knowledge to explore new ideas.

What do microworlds reward?

Let's look at what microworlds reward in their users. To understand this, we should look at the Papertian public entities that would be constructed within these microworlds.

Consider Minecraft. Fans of the TV and book series *Game of Thrones* have built a version of Westeros, the series' fantasy realm, from 1.2 billion bricks. Compared to the size of the characters, this Minecraft construction is the size of Los Angeles{westeroscraft}.

Creating something the size of Los Angeles, even if it only exists in cyberspace, requires considerable skill. Builders need to be able to place single blocks to form superstructures, and compose those superstructures to form hyperstructures, and so on{reddit}.

When shared on servers, Minecraft artifacts are public entities. How are these evaluated by other members of the public? Like other works of art. It appears that complexity for its own sake is not admired; non-trivial complexity must be twinned with aesthetic beauty for an artifact to be admired{mash}.

The same can be said of the creative writing and musical jamming microworlds. Ralph Ellison, the American novelist, has said: "Good fiction is made of what is real, and reality is difficult to come by," which can be interpreted as saying that fiction is judged on non-trivial complexity ("difficult to come by") and truth value instead of aesthetic beauty.

<< MUSIC - better when blends well >>

Flappy Bird, on the other hand, rewards reflexes. A public entity in the Flappy Bird framework is a player's point score. Using this is a feedback mechanism does increase your reflexes, in a very narrow way, but does not improve much else.

Why do these rewards lead to learning?

The feedback loops of the microworlds mentioned should do the following: * Making things in Minecraft makes you better at building complex structures. * Creative writing makes you better at expressing yourself in writing, and writing the "truth". * Jamming makes you better at blending musical styles.

We see this happen when AIs learn using Machine Learning techniques...

{dewey} The Child & The Curriculum, John Dewey {ackermann} http://learning.media.mit.edu/content/publications/EA.Piaget%20_%20Papert.pdf {convsinst} http://papert.org/articles/const_inst/const_inst1.html {situconst} http://www.papert.org/articles/SituatingConstructionism.html {westeroscraft} http://www.wired.com/2013/03/westeroscraft-game-thrones-minecraft/all/ {mash} http://mashable.com/2013/02/13/amazing-minecraft-creations/#gallery/25-minecraft-creations-that-will-blow-your-mind/53111bee12d2cd0acb004f80 {reddit} http://www.reddit.com/r/Minecraft/comments/dibqq

1.4 Microworlds are engaging

What is the key to motivation?

According to Dan Pink, it's autonomy. Roman Krznaric agrees, and uses the following statistic to prove his point: 47% of self-employed people say they are "very satisfied" with their jobs, compared to only 17 percent of those in regular employment{krznaric}.

Here is the table from which he got his data{joywork}:

% of respondents Full-time Part-time Self-employed Very dissatisfied 5.8 3.2 1.6 Dissatisfied 11.2 10.5 4.7 Neutral 18.7 18.9 12.5 Satisfied 46.9 48.4 34.4 Very satisfied 17.3 18.9 46.9

There are more than twice as many dissatisfied people working full time compared to those who are self-employed.

Why would this be?

The report has this to say:

> This could be attributed to the control that the self-employed have over their work: whilst many work very long hours, their ability to determine when, where and how they work may contribute to their high levels of satisfaction.

If you have the "ability to determing when, where and how" you work, that means you have autonomy over your work.

Do microworlds provide autonomy?

The first stage of our microworld-defining loop is coming up with an **idea** that obeys the constraints of the *structure* of that microworld.

Imagine you are a self-employed, freelance web designer. You have been tasked with coming up with a new landing page for a coffee shop brand.

You have autonomy over when you work, where you work, and how you work. You can work only between 22.00 and 04.00. You can work from home. You can use or skeumorphic design, or flat design.

Let's say you try skeumorphic design, but after you **implement** some skeumorphic elements, and **evaluate** them, something seems off. You update your **ideas** and try flat design instead. Much better.

You exercised your autonomy in *how* you work.

But however much you iterate, you still need to deliver something that looks like a web page. You can't deliver a design for a printed book. Or a design for a 15-minute movie.

Those are the constraints of the *structure* of your web design microworld.

What would a microworld look like without autonomy?

Let's imagine our web design microworld without autonomy.

Your task is still the same: creating a landing page for a coffee shop brand.

But this time, you must work between 09.00 and 17.00 on weekdays. You must work from the office, from your cubicle. You're free to use skeumorphic design or flat design... but your boss *really* likes skeumorphic design.

After **implementing** your initial skeumorphic design **idea**, you **evaluate** it, and as before, you find it lacking.

Unfortunately, you can't change it. Because your boss likes it, and their opinion overrules yours.

The structural constraints of the microworld still apply. But now some arbitrary constraints *also* apply: over where, when and how you work.

These additional, arbitrary constraints suck the fun right out of the microworld. They make the microworld less engaging. They make it feel like work, not a game.

What does engagement lead to?

When a microworld allows for autonomy, it is a highly motivating place to experiment in.

In Minecraft, this motivation leads to people building their first basic block structure. For a band jamming together, or a novice writer, it might lead to the first work they're comfortable sharing with friends. For the budding programmer, it usually leads to their first non-trivial, mostly-working program.

And that leads to a sense of accomplishment. But that's not where it ends.

After an initial success, the temptation is there to tweak the original, ever so slightly. The first block structure leads to a house; one song, story or program leads to another, similar in structure but differing in detail.

Eventually, we end up with a blockrealm the size of Los Angeles; a band and author with a string of hits; a programmer in charge of a program that serves billions around the world.

As we argued in the previous section, each iteration of our loop led to learning for all involved.

But it took the engagement enabled by autonomy to start the loop spinning in the first place.

[pos = H, center, botcap]ll {krzn aric} How To Find
{joyw Fulfilling Work ork}
http://twfold.theworkfoundation.com/assets/docs/publications/145_Joy_

1.5 Virtual microworlds

Unlike our jamming and creative writing microworlds, Minecraft is a *virtual* microworld. This means that software is an integral part of the microworld, in a way that is not necessarily true for mostly physical microworlds.

How does software change things? It eliminates routine tasks, by delegating those to the machine. Let's see how that works in practice:

The physical microworld equivalent of Minecraft is a pile of Lego bricks on the floor. Unlike Minecraft, everything is, understandably, manual. The only way to search for certain kinds of bricks is by conducting a linear search, or, if you're very organised, sorting the bricks first.

There is a limit to the number of bricks the average person can handle. This limits the complexity of our Lego creations. There are also constraints of time and space.

Playing with Lego can occupy the entire floor. This means that only those people with access to the floor can play. Even at the most social Lego sessions, it's unlikely more than a handful of people will be able to participate simultaneously.

And before long, the Lego will have to be tidied away. As bricks are physical, this usually entails breaking up any structures and putting single bricks back in the box.

Minecraft is different. As the bricks are virtual, they can be sorted and organised by the machine. They can be practically infinite in number, and the increasing complexity in structures can be handled by the user interface treating a grouping of blocks as one.

The environment is virtual too, so bricks don't have to be tidied away when you're done; their patterns are stored in bits until you're ready to return. This makes it easier to work on larger-scale projects, as it eliminates the need to ever start over from scratch.

It also makes it easier for users to collaborate. Builders no longer have to be friends, or even geographically close to each other, to work together in the same environment. They just need to be logged in to the same server. This encourages large-scale collaborative projects.

But software also creates a barrier to entry. To participate in a virtual microworld, you need specialised equipment - a computer - that can run it. That said, you need specialised equipment, such as musical instruments, to participate in physical microworlds too, and it seems that computers are becoming less and less "specialised" as time goes on.

However, to run a virtual microworld, you also need specialised *software* to go along with your hardware. For Minecraft, you need to install the Minecraft software package. For programming microworlds, you often need to install an interpreter, configure an editor and to set up a build system.

The easier a microworld is to enter, the greater its potential audience. The microworlds that are easiest to enter require the least set up of specialised software on the part of the user.

The web browser is fast becoming less and less specialised. All computers come with them pre-installed, and most users spend a significant part of their computer time using them.

Microworlds in the web browser are arguably the most easily accessible virtual microworlds. Let's take a look at some current examples.

CodePen (http://codepen.io/)

CodePen is a browser-based virtual microworld for front-end web development, i.e. HTML, CSS and JavaScript. The creation view looks like this:

<SCREENSHOT>

There are four panes: one each for HTML, CSS and JavaScript code, and one that dynamically renders the result.

This microworld leverages the browsers capacity for dynamic rendering: of parsing and showing the results of code on the fly. A website allows you to experiment with creating websites.

Created entities in CodePen - "Pens" - are public entities and can be browsed by others, and even tweaked by them. This creates a rich environment for creativity and experimentation, and as we have argued so far, a rich source of learning, in this case in the domain of front-end web technology.

Pacemaker

Blogging systems (Tumblr / Medium / WordPress etc)

Chapter 2

Implementation

2.1 Talking to Machines

After researching microworlds and developing my theoretical framework, I felt ready to design my own microworld, which I named *Talking to Machines*.

Concretely, this entailed designing an environment in which an idea-implementation-evaluation loop could be carried out. For the reasons discussed in the previous section, I decided to host this environment within the web browser.

This decision brought with it some constraints: the interface would have to be built from HTML and CSS, and interactivity would have to either be supplied by client-side JavaScript, or a server-side evaluation environment. But this decision also enabled the increased accessibility discussed earlier.

I decided to focus this microworld on programming, as that is an activity I have a lot of experience with. In a web environment, that would entail using JavaScript as the programming language, as that is available on the client-side, and removes the need to execute a different language on the server and then send back the results.

However, I'm not a great fan of JavaScript when it comes to teaching beginners to program (see here for why{wat}). Instead, I chose to use a dialect of Lisp - Clojure (in its JavaScript-hosted form, ClojureScript) - as Lisp has a history of being a language well-suited for beginners (MIT used it in an introductory programming course for years{sicp}).

Evaluating ClojureScript forms in the browser (warning: technical)

At the time of creating *Talking to Machines*, there was no way to dynamically evaluate ClojureScript in the browser, as the reader and compiler for ClojureScript forms was in fact a Clojure program that could only run on the JVM.

The following makes this rather complicated concept a little easier to understand:

The ClojureScript syntax for printing "Hello World!" to the console looks like this: > (.log js/console "Hello World!")

If ClojureScript could be evaluated in the browser, you'd expect to be able to do something like this, in *JavaScript*: > ClojureScript.eval('(.log js/console "Hello World!")') and have "Hello World!" printed to the console.

Unfortunately, in standard ClojureScript, you can't do that (at the time of creating *Talking to Machines*).

Instead, a *Clojure* program, running on the JVM (i.e. certainly not in the browser), has to *transpile* a ClojureScript file to a JavaScript file, i.e. > (.log js/console "Hello World!") in `foo.cljs` becomes
> console.log("Hello World!") in the output file `foo.js`, and `foo.js` *is what is evaluated in the browser.*

All this means that > ClojureScript.eval('(.log js/console "Hello World!")') doesn't happen in JavaScript. Instead, we have: > (emit (read '(.log js/console

“Hello World!”)’)) => ‘console.log(“Hello World!”)’ in server-side Clojure, and > eval(‘console.log(“Hello World!”)’) in the browser (that’s standard JavaScript `eval` there, no funny business).

So what does this mess actually mean, practically?

It means that if a user writes ClojureScript in the browser, we won’t be able to evaluate what they wrote without having a backend Clojure process transpiling their code to JavaScript for us.

This isn’t actually a massive issue, as we still don’t evaluate arbitrary code on the back-end, which would obviously be a Big Deal (we just transpile it instead), but it would introduce some network lag between the **implementation** and **evaluation** stages of our microworld loop.

And as one of the points of my thesis is that the tighter the loop, the more engaging the experience, this was something I wanted to avoid if at all possible.

Kanaka to the rescue

Luckily for me, there was a fork of ClojureScript that did allow evaluation in the browser, written by a developer called Joel Martin (Kanaka on Github). At the time of writing it is still a fork, and hasn’t been merged with ClojureScript core, probably because it contains “miscellaneous broken things that have not been tracked down yet”.

But it was definitely good enough to evaluate ClojureScript for a microworld, and it probably took me less time to modify Martin’s code to provide my wished-for `ClojureScript.eval` JavaScript function than it took me to explain why that was even necessary.

So why did I go to all this trouble?

This might seem like a very convoluted mess to get into just to avoid writing pure JavaScript, which, after all, is good enough for sites like Codecademy. That’s a fair point but I believe that the declarative purity of ClojureScript forms makes up for the chaos going on behind the scenes. As you will soon see.

A better first impression of programming

Most very-first-introductions to a programming language, or programming in general, involve printing the text “Hello World!” to the screen. Later exercises usually include things like printing the numbers “1 2 3 4 5 6 7 8 9 10”, or adding up all integers less than 100.

Though these feats may have been impressive 30 years ago (and might hold some intellectual curiosity today), today these things fail to blow anyone’s socks off.

Producing plaintext is far from unimportant (most webservers do little else), but in a world where technology means Oculus Rift and 4K Netflix, it’s not really sexy.

Is there something simple we can have beginners do, to explain functions and variables and loops, that is more interesting than printing text to the screen?

We could try the activity that captivates children (and artists) all over the world: drawing things. With colours!

Declarative shapes

The most basic form of computation is a single function call. It is the simplest action that *does* something (variable assignment does something too, but indirectly, setting things up for later function calls).

It makes sense that drawing a shape to a screen should be the result of just one function call, instead of requiring a novice to construct a class, call methods on that class, and then draw that class to the screen after having acquired a graphics context.

The more declarative this function call, the easier it is for a novice to deduce its meaning from what happens when its evaluated.

Consider: > (circle 50% 50% (radius 50) (fill :red))

To a non-programmer (and perhaps even a programmer unused to S-expressions), this line of text looks alien. It conforms to a very different kind of grammar than what we're used to reading.

This is why, when I ask non-programmers what they think the "50" in the line above represents, they often have no idea. The text as a whole is just too strange for them to parse and guess that the proximity of "radius" to "50", and the brackets that enclose them, means that the two tokens have something to do with each other.

If we were in a static environment, the only way to learn about what the above text does would be to read about the syntax and grammar of Lisp. This is often how programming is approached, from a mathematical perspective. The only way to find out if you had a correct understanding of Lisp syntax and grammar would be to ask a teacher. The gap between **implementation** and **evaluation** would be so large that it would kill a lot of your motivation.

Fortunately, *Talking to Machines* is a microworld. This means that a student is able to change the above text to: > (circle 50% 50% (radius 100) (fill :red))

As soon as this happens, the text is **evaluated**, and draws a *bigger* circle on the screen. Immediately, the student realises that the number that was "50" but is now "100" somehow affects the *size* of the drawn circle.

Through their interactions, the student has **constructed** this knowledge for themselves. According to our constructionist learning theory, this knowledge should be far more effective than being *told* that the `(radius 50)` form controls the radius of the drawn circle.

Looping in *Talking to Machines*

Let's see how *Talking to Machines* supports our microworld loop.

Once a student is somewhat familiar with how the microworld works, they can come up with the following **ideas**: * I want to draw a red circle * Can I make it blue? * Can I make it bigger? * Can I move it around? * Can I draw more than one? * Can I draw other shapes? * Can I animate them? * etc.

Their **implementations** of these ideas take the form of composed function calls. These function calls are immediately **evaluated**, and the student sees visually how their change of code affected the change in what was drawn to the screen.

For example, the first idea above, of drawing a red circle, looks like this: > (circle 50% 50% (radius 100) (fill :red))

Making it blue looks like this: > (circle 50% 50% (radius 100) (fill :blue))

Making it bigger: > (circle 50% 50% (radius 150) (fill :blue))

Moving it around: > (circle 25% 50% (radius 150) (fill :blue))

As we can see, the student iterates the loop: 1. Hmm, what does this do? (idea) 2. I'll tweak it! (implementation) 3. Ah, that's what that does! (evaluation) 4. Hmm, but what about *this*? (loop) and iterations happen *extremely fast*. This keeps motivation and engagement high.

The student learns something on every iteration, which feels rewarding, which pushes them to iterate again.

Limitations of this microworld

Talking to Machines was a success on a *structural* level, but a failure on a *practical* level, at least given the time constraints of this thesis.

It successfully enabled the learning iterations of a microworld, but it required too much prior knowledge of students for me to be able to conduct workshops with many students at once.

Talking to Machines offered learning about two topics: the syntax of a programming language, and hands-on knowledge of computation. These are distinct: a programmer can know that she wants to write a recursive loop, but not know the syntax for this in JavaScript.

Hands-on knowledge about computation is hard to acquire without knowing a little syntax in at least one programming language. This is because if the **implementation** of your computational **idea** (such as how to print only odd integers less than 100) can only be expressed in pseudocode, you will only be able to **evaluate** your idea by writing it out by hand or asking a teacher if you are correct.

Being able to write in the syntax of a programming language means being able to **implement** your computational **ideas** in a form that can be instantly **evaluated** by a machine. On the bright side, this means that you can learn things in an effective, engaging way without having to ask for the permission or judgement of a human teacher or academic institution.

The downside is that you need to learn that syntax before you can reap the rewards of machine evaluation. It is possible for students of *Talking to Machines*

to learn Lisp syntax by trial and error, by chopping and changing parentheses until the code evaluates. But this is not the most efficient way to learn syntax, so engagement suffers as a result.

Instead, students could be guided through their first attempts to produce syntactically correct code, for example: > A Lisp function call has the form (f x) where f is a function and x is the argument. Try calling the "circle" function with the argument "50"!

This kind of guidance takes time to do well, either by employing many human assistants with the skill to teach students about syntax, or by programming a tutorial in which the computer itself guides the student.

I didn't have this kind of time. Ideally, I wanted to explore the knowledge-constructing effects of microworlds on students, without having to teach them a new syntax beforehand.

This lead me to create a microworld I called *Narrative Roulette.*

{wat} https://www.destroyallsoftware.com/talks/wat {sicp} http://mitpress.mit.edu/sicp/course.html

2.2 Narrative Roulette at Kungsholmens Gymnasium

Narrative Roulette (NR) is a microworld that does not require students to learn a new language before being able to fully explore it. Instead of being built around a programming language like Lisp, *Narrative Roulette* is built around a natural language: English.

A round of *Narrative Roulette* goes like this: 1. The teacher selects an interesting perspective for students to take. For example: "You are the captain of a sinking ship". 2. Students have 15 minutes to write a few paragraphs from that perspective. Submissions are anonymous. 3. After those 15 minutes are up, students read each other's submissions and discuss them.

The benefits of being virtual

As we can see, NR is a creative writing microworld. However, it is also virtual and browser-based. Students read and write their texts in the web browser, and discussions are had while reading texts off screens.

This allows the software to do all of the clerical work. Submissions are edited by students using a web frontend, and then accepted and filed by the backend. As the frontend does not send any identifying information to the backend, anonymity is guaranteed. When it comes to letting students read each other's work, the backend takes care of sending a copy of each text to each student that requests one.

Without this custom-built software, students would have to write by hand, and hope their handwriting is not recognisable by their peers or their teacher. The teacher would have to accept submissions on paper, photocopy them, and then hand them out to the class. This would waste both paper and time, and result in lower motivation for the students.

Narrative Roulette as a Microworld

Let's look at this round definition compared to an iteration of the microworld loop: 1. Students come up with **ideas** of what it would like to be the person in the situation named by the perspective. 2. Students **implement** their ideas in fictional texts. 3. Students **evaluate** those texts by hearing what their peers think of them.

Narrative Roulette is more directed than *Talking to Machines* (TTM). TTM was completely open-ended, allowing the student to draw whatever they wished, whereas NR asks you to write from a particular perspective.

This does not seem to have been a problem; in fact, it seems to have helped focus the imaginations of the students taking part. A totally blank canvas can be intimidating. A frame, such as "you are the captain of a sinking ship", can help you start by giving you a concrete environment to imagine and react to.

How do you explore this microworld?

Let's see how this works in practice.

A narrative in *Narrative Roulette* contains characters, actions and reactions. Narratives are written from the first-person perspective, so a major character is the narrator, who is the character defined by the round.

Texts usually take the form of internal monologue. This gives the reader first-hand insight into the mental processes of the main character. It also allows the writer to identify, examine and play around with the mental processes they believe the main character could be having.

These mental processes lead to actions, or actually the narrator's description of their actions. For example, the captain of a sinking ship can decide to tell her crew that they need to get into a lifeboat. From our vantage point, inside the narrator's mind, we can see *why* she does this: is she trying to help everyone on board, or is she getting herself and the crew off the ship and leaving the passengers to die? The writer gets to decide exactly how the narrator's thoughts manifest in actions.

Consequences in the narrative are also chosen by the writer. Characters react to the actions of the main character. The writer has to decide what those reactions are; how they manifest in the actions of those other characters; how the main character *infers* these reactions from the other characters' actions; and finally how the main character reacts to all of these consequences.

And then there are the writer-defined constraints of the storyworld. When we imagine a sinking ship, most of us would imagine enough lifeboats to evacuate all the passengers and the crew. We are in control of the storyworld, so why wouldn't we dream up circumstances in which everyone gets saved?

Because in the real world, things don't always work out so nicely. It's quite possible that there will be fewer seats on the lifeboats than passengers and crew on the boat. Writers who want to challenge themselves will realise this, and in their storyworlds, some people will be left behind. The ensuing moral dilemma will be interesting to both read and write about.

In essence, this is the difference between reading and writing text. When you write, you have to make choices. You choose who your characters are, how they behave and what consequences they have to deal with. Not only this, but you also have to choose how you want to convey all of this to the reader. When time and space is limited, as in *Narrative Roulette*, the best strategy is to say as much as possible using the fewest words.

To recap, writers explore the *Narrative Roulette* microworld by **implementing** their **ideas** about characters and situations in English prose.

Why is this worth doing?

Narrative theorists have said that the reason we read fiction is to gain a better understanding of the psychologies of the main characters, from their description in the text{zunshine}. We read to learn about why people act the way they do,

by analysing fictitious characters and situations. This exercises our Theory of Mind - our capacity to intuit what other people are thinking and how they will behave.

Reading fiction is a worthwhile activity because the more accurate our Theory of Mind, the better we are at understanding and predicting the thoughts, feelings and behaviour of others, and even ourselves.

When we *write* fiction, we have to **construct** characters, situations, actions, reactions, dialogue and description. In each of these areas, we come up with **ideas**, **implement** them in words, and then **evaluate** them by reading back our text as we are writing it.

According to the constructionist learning theory, this means that we learn about characters, situations, actions, reactions, dialogue and description. Learning how to write rounded characters means learning about human psychology. Learning how to write characters' behaviour means learning about human behaviour. Learning about how to construct interesting situations means understanding the interplay between people and their social, physical, cultural and emotional environment, and being able to manipulate that interplay in the most interesting way.

Writing fiction is an exercise in constructing psychologies, situating them in a storyworld, and then putting them through a chain of events that make up a narrative. This means that a creative writing microworld rewards (and thus improves) theory of mind as well as intuitive theories of causation.

The very best fiction involves realistic characters dealing with realistic consequences. Creative writing microworlds help students construct knowledge about human nature and the world we live in.

How are these texts evaluated?

Criticising fictional texts is hard, we can see that much from the disagreement that often surrounds literary criticism. One person's favourite character might be the least favourite of another, and so on. This makes it difficult for a single teacher, or any single grade-setter, to pass judgement on a work of fiction.

In *Narrative Roulette*, the entire readership of a work passes judgement on a work. Currently, they do this by reading each other's work in groups, and discussing them. The software can track how many times a work has been read. In the future, the software will allow readers to signal that they like a work, and to share it.

This allows for many axes on which works can be evaluated. A work of fiction that no-one reads is a failure on a functional level - texts exist to be read, so if no-one reads a text, it failed at what it was meant to do. We could call texts that no-one reads *valueless*.

Some texts will cause controversy. This means that the texts are read and then fiercely debated. Sometimes, debaters will not approve of the text. But the fact that the debaters read it, and then found things in the text worthy of discussion, means that the text has some value.

Other texts will be liked by the majority of readers. This does not necessarily make them *better* than the more controversial works. But universal reader approval is still a form of success, as it means that people read the work. As with the controversial scenario, the fact that people read the work, and found things in the text worthy of liking, means the text has value.

In this way, texts are primarily evaluated on their *readability*, by the reading behaviour of other students - the writer's peers. This can be called a *structural* evaluation of a student's writing: how good they are at communicating their ideas through the medium of language.

The *content* of a student's writing is evaluated by how much discussion it generates, and how much people are willing to share it. Review-style comments ("I liked this part, but not this part") count as discussion.

This means that the traditional way of evaluating student's texts, by a teacher who writes what is in effect a review (with an accompanying grade), is just one component in a work's overall evaluation in *Narrative Roulette*. This makes a teacher's (or grade-setter's) opinion less important. If a teacher considers a work of low value, but the writer's peers all read and share it, the work will have high value *despite what the teacher thinks*.

This makes evaluation more democratic, and more like real-life. After leaving school, a teacher or school's opinion of work loses the power to negatively affect a student's future: "If you don't write your essay using this exact structure, you'll get a bad grade, and not be able to go to the university you want to." But after leaving school is exactly when the opinions of peers becomes all-important: "If people don't read your articles (or memos), you won't have a job at this company." Which means that *Narrative Roulette*'s form of evaluation is better preparation for life than that of traditional school.

How Anonymity Helps

But if the behaviour of peers dictates the evaluation of a work, won't evaluation descend into a popularity contest? Won't students only read, discuss and share their friends' work, regardless of its actual quality?

They can only do that if they know who wrote a work. Texts in *Narrative Roulette* are anonymous, as the front-end of the software sends no author-identifying information to the back-end.

This forces students to evaluate a text on the strength of only the text itself. It removes bias, unconscious or otherwise. Readers cannot discriminate based on their relationship to the writer, or by the writer's age or gender. They can only discriminate by the writer's abilities, which is entirely the point of the exercise.

This kind of anonymity is rare if not entirely unseen in a classroom setting. As explained above, it brings with it great benefits. Fiction already has the capacity for enabling discussion without condemnation, such as in Nabokov's portrayal of paedophilia in *Lolita*. Authorial anonymity increases this capacity, and allows students to discuss things without fear of being censored by their teachers or their peers.

And this kind of anonymity would not be possible (or at least practical) without technology.

Limitations of Narrative Roulette

- would have been great to have more time
- so... let's try and put it online!

[pos = H, center, botcap]ll {zunshine} Lisa Zunshine, Why We Read Fiction: Theory of Mind and the Novel

2.3 Narrative Roulette on the Web

A problem with *Narrative Roulette* was that it required a classroom and lesson-time for students to take part.

I managed to obtain a classroom without taking time from students on one occasion, by holding a voluntary *Narrative Roulette* workshop on 21 January 2014 at Kungsholmens Gymnasium. Turnout was better than expected (15 students and friends attended), and the session proved that the workshop would work. But it wasn't a true proof of the concept until it was tried on students that *had* to be there as part of their ordinary school day.

Finding lesson-time for *Narrative Roulette* workshops was extremely difficult. Teachers, understandably, seemed to prefer to spend their lesson time achieving the goals of their curricula directly, with their own lesson plans, rather than taking a risk on the indirect value provided by *Narrative Roulette*.

A potential solution to this problem was to remove the dependency on the physical classroom, and try to take a share of students' leisure time instead of their school-time.

NarrativeRoulette.com

I registered the domain http://narrativeroulette.com and modified *Narrative Roulette* so that rounds could take place online.

The modifications were the following: * Now, rounds would last for one week instead of 15 minutes, with submissions accepted at any time that week. * All submissions were anonymously posted to Facebook, by the Narrative Roulette Facebook Page (https://www.facebook.com/narrativeroulette). * Discussion took place in Facebook comments. * Stories could be shared via Facebook. * Communication between myself and Narrative Roulette participants took place through the Narrative Roulette Facebook page.

Is this still a microworld?

An iteration of the microworld loop in NarrativeRoulette.com looks like this: 1. The user reads the perspective of the current round. For example: "You are female. You are more scared than you've ever been in your life." 2. The user writes a short narrative, from that perspective, using the editor at NarrativeRoulette.com. This is the **idea** stage of the microworld loop. 3. The user submits their narrative. This is the final **implementation** stage of the microworld loop. 4. People who have liked Narrative Roulette on Facebook see the user's text published by Narrative Roulette. They **evaluate** the text by liking, sharing and commenting on it, all in Facebook.

As we can see, in the NarrativeRoulette.com microworld, the **idea** and **implementation** stages of the microworld loop stayed the same as in the classroom version. The changes were to the **evaluation** stage, where texts were read, discussed and evaluated on Facebook - a virtual space instead of a physical one.

How did this work in practice?

The NarrativeRoulette.com microworld was far less engaging than the classroom version. A round on NarrativeRoulette.com received an average of 7 submissions, while a round in the classroom version received as many submissions as students in the classroom: around 30. This was despite the reach of Narrative Roulette's Facebook page being 38 people.

I believe that there was also an indirect change to the **idea** stage, and this is what made the difference between the engaging power of the two versions.

When you're in a classroom, attending a *Narrative Roulette* workshop, you have a clear goal: to write a narrative from a certain perspective within a short timespan (15 minutes). You are motivated to do this, even if no teacher keeps an eye on your screen, because you know your peers are all doing this at the same time, and you want to add to the work being created.

Online however, the communal aspect of writing is lost, together with much of the earlier motivation. Now you're not writing because you have to, or for your classmates; you're writing for the internet, because you *want* to. And you don't have to submit in the next fifteen minutes, you can submit any time until the end of the week.

The fun of evaluating texts seemed to disappear too. In the classroom, readers didn't know exactly who had written a text, but they knew it was someone sitting in that room. I believe that lead them to be more forgiving, knowing that the writer was one of their peers.

When an anonymous submission appeared on Facebook, readers had no idea who had written it. I believe this lead them to be less forgiving, which meant that they were reluctant to share, like or comment on the texts.

NarrativeRoulette.com takeaways

Moving *Narrative Roulette* online removed the need for acquiring classroom space and lesson time from students. However, it failed to be as engaging - there was less work produced and less discussion generated than *Narrative Roulette*'s physical version.

This lack of engagement was perhaps due to the loss of the classroom. Without the classroom's atmosphere of authority, it was hard to motivate students to participate in the microworld.

The content of the microworld doesn't seem to be problem, as engagement levels amongst the students for the classroom version was reportedly extremely high{ingulfson}. Instead, the problem seemed to be the transition from a physical to a virtual space, especially for the **evaluation** aspects of the microworld.

[pos = H, center, botcap]ll This all means that NarrativeRoulette.com is not worth pursuing as a project in its current form; the physical version of *Narrative Roulette* seems to be superior. {ingulfson} Personal correspondence with Eileen Ingulfson, teacher at Kungsholmens Gymnasium.

Chapter 3

Evaluation

3.1 What does this mean for education?

Talking to Machines was a microworld. We saw from its structure how it could enable learning, but we decided it was impractical to test in a classroom setting. *Narrative Roulette* kept the same structure, but changed domain from programming languages to natural language, to make the microworld more accessible.

Narrative Roulette was engaging: there were <X> submissions by <Y> students in three obligatory sessions at Kungsholmens Gymnasium. This means that <Y> students iterated the microworld loop *at least* <X> times.

Why? While writing, students must iterate the loop at a micro-level in their own heads (or on their own screens): they have ideas; implement them; read them back; and then update their ideas. The lower bound for a student's number of iterations of the loop is 1 per submission. This occurs if they do *everything right the first time*. In that case, they have an idea, implement it, and evaluate it to be perfect; impossible to improve. Then they send in their submission. Given this, it makes sense to assume most students iterate the loop multiple times per submission.

Did iterating the loop lead to learning?

<< INSERT EVIDENCE OF INSIGHT, FROM SUBMISSIONS, HERE >>

Microworlds vs Traditional Learning

Let's compare the structure of traditional teaching methods with microworlds. Traditional teaching is often split into three activities: lectures, exercise sessions, and exams. Lectures provide students with theory, exercise sessions (or seminars) allow students to put that theory into practice by solving problems, and exams allow students to demonstrate both their theoretical and practical skills, in isolation - test conditions, generally with no help from textbooks, the internet or their peers.

There are rough parallels between these methods and microworlds. Lectures correspond to **ideas** in a certain domain. A major difference between ideas in lectures and ideas in microworlds is that ideas in lectures are *external*. They are other people's ideas, such as "Recursive loops never terminate if you leave out a base-case". In a microworld, the ideas you have are *internal*, your own: "If I write this base-case, my loop will terminate. Oh, I was wrong".

Exercises correspond to **implementation**. However, unlike microworlds, traditional exercise sessions have students implementing external ideas. For example: "This is Pythagoras' theorem, now apply it to this scenario!". In a microworld, it is a student's internal ideas that are implemented: "I wonder if I can use Pythagoras' theorem to help me draw a right-angled triangle."

Exams are the **evaluation** stage. Unlike microworlds, exams do not give instant feedback. Failing at problem-solving in an exam is not a learning experience, it

results in a period of uncertainty, which ends leaving you with a lower grade. If having your exam marked by a teacher taught you something - perhaps how to solve a problem correctly - after the course is over, you have no way to apply that knowledge, or to prove that you now have it.

When you fail at problem solving in a microworld, you know immediately, enabling you to update your understanding immediately. Evaluation is *productive* in a microworld, in a way that exams generally aren't.

In short, traditional learning methods do provide **ideas**, **implementation** and **evaluation**, but in components that are spread out in time. A microworld places these components in a tight loop that is iterated many times at speed. It is for this reason that microworlds lead to more learning than traditional teaching methods.

Agile methods vs The Waterfall Model

There are echoes of this comparison in Software Engineering ideology. Traditional teaching methods arguably correspond with The Waterfall Method. The Waterfall Method consists of{wiki:water}: 1. Requirements and Design (Microworld **idea**s). 2. Implementation (Microworld **implementation** , obviously). 3. Verification (Microworld **evaluation**).

As in traditional teaching, these steps are separated in time. The requirements and software design are done first, before any implementation. Then the implementation is carried out. Finally, the implementation is verified by the customer.

According to Dean Leffingwell and many others, "the Waterfall Model doesn't work"{leffingwell}. Instead, most modern software engineers prefer agile methods.

These agile methods move the requirements, design, implementation and verification steps closer in time, and prefer fast iterations. Agile methods make software engineering into a microworld.

Are there schools teaching this now?

Berghs School of Communication

Berghs School of Communication uses Action Based Learning, a philosophy that is extremely similar to microworlds as defined in this paper. http://www.berghs.se/content/berghs-pedagogik

Hyper Island.

Hyper Island use "constructivist methodology". http://www.hyperisland.com/singapore/sgma/philosophy

How can the peer-evaluation of Narrative Roulette be expanded to other microworlds?

Works were evaluated in *Narrative Roulette* by a student's peers, based on number of readers, and those readers' sharing and discussion behaviour.

Can this be extended to other microworlds?

The only domain-specific detail in the above evaluation method is "reading". This can be generalised to "consumed", or "used" for other microworlds. For example, in a music-jamming microworld, we can measure how many of a student's peers listen to produced songs; in a programming microworld we can count how many other students interact at length with a program. The sharing and discussing behaviour remain the same.

A constraint is that only "projects" can be evaluated in this way. This method is better for a consumable product than for the solutions to a problem set. But in fact this is entirely intentional: it forces us to use microworlds to enable students to create "public entities", which was a defining feature of constructivism.

Should we drop lectures, exercise sessions and exams?

This doesn't necessarily mean we should drop traditional learning methods. Lectures and textbooks are still a very valuable source of knowledge, and they are not at odds with the microworld paradigm.

Lectures and textbooks can be combined with microworlds when students take part in **self-directed research**. This is what happens when a student discovers a gap in their knowledge when exploring a microworld. For example, in *Talking to Machines*, a student might want to learn about recursive loops, in order to draw concentric circles. They could then read a chapter about recursive loops in a textbook, or watch a lecture about them.

The difference from traditional teaching methods is *why* a student reads a book or watches a lecture. Now, the student is *actively* seeking information, instead of having it pushed at them and being told to remember it. They can apply what they learn out of their own choice, in their own way. Until a student does this, they have not *internalised* knowledge - made it their own.

Exercise sessions are subsumed by microworld exploration: a student practices **implementation** every time they iterate the microworld loop, so they don't need to work on practicing implementation in isolation from having **ideas** and **evaluating** their work.

Traditional examinations could be entirely superseded by the peer-based evaluation discussed above, but it's possible such a dramatic change in the structure of education would be met with resistance. This is the focus of the next section.

[pos = H, center, botcap]ll {wiki :water} http://en.wikipedia.org/wiki/Waterfall_model
{leff ingwell} Dean Leffingwell, Scaling Software Agility: Best Practices for

Large Enterprises

3.2 What problems are there with microworlds?

Let's imagine that microworlds - virtual, physical and hybrid - are adopted in the mainstream curricula of schools and universities worldwide. What consequences would this have for the structure of education?

Fully embracing microworlds

As discussed in the previous section, lectures could remain relatively unchanged, with the difference being that they are no longer mandatory, but can be voluntarily attended by students undertaking self-directed research. These lectures could still be divided by subject in the same way as today, for example: the biology of psychopathy is a natural science; the ethics of violence is a social science; and the literature of crime is art. The reasoning behind this is that the subject division simply dictates the *angle* the subject will be approached from. As lectures are voluntary, students can mix and match these angles as they wish.

Instead of producing work within a subject area such as Biology, Philosophy or English Literature, students would then apply their knowledge by exploring microworlds that span multiple subject areas.

Their aim would be to create works that can be peer-evaluated, by the consumption of the work by a student's peers, and the sharing and discussion it generates. A possible work could be creating an interactive virtual environment, like a game, where players explore the concept of psychopathy, from a biological, ethical and literary perspective.

This could be something like a point-and-click adventure where you play a psychopath, and perceive the world as a student believes a psychopath would. The student who built the game would have to study biological, psychological and literary work on psychopathy to create a realistic world. The student could then explore ethical questions by offering players choices about how to behave. The resulting narrative in the game world would demonstrate students' literary abilities.

For variety, students could work on multiple projects in multiple microworlds, all with different themes, at the same time during a term. Every so often, a project would come to an end and students' work would be presented in an exhibition, and evaluated based on the reaction of peers to a student's work.

After one set of projects have been evaluated, a new set of projects, in their own specifically-designed microworld, would begin.

From lesson plans to microworld construction

If lessons are replaced with microworlds, those microworlds will have to be constructed. Physical microworlds, like jamming with instruments or creative writing on paper can be constructed by teachers today.

Virtual or hybrid microworlds are harder to construct without technical skills. *Talking to Machines* and *Narrative Roulette* had to be programmed specifically,

they had too many custom requirements to be put together from a standard system.

I believe I gained the majority of the skills required to program those microworlds from working in the software industry, not from my Computer Science education. This means that it takes software industry experience to construct microworlds - something which the vast majority of teachers do not currently have.

Even if the construction of microworlds were outsourced to the software industry, teachers would need considerable experience with those microworlds to help and advise students on how to best explore them. As most microworlds are relatively complex software constructs, this is no easy task.

Transfer of control

Allowing students the freedom to have their own ideas and to choose their modes of expression transfers a lot of control from teacher (or school) to the student. If a teacher's performance is rated by the grades of her students, that teacher may be understandably reluctant to reduce her amount of control, as losing control over something means increasing risk.

The same applies for politicians. If the performance of a department of education is valued by global ratings, such as the PISA rankings, that department will be understandably reluctant to transfer control to teachers (who would then transfer control to students, according to the microworld model). So instead, governments push for more exams, more grades and more inspections of educational institutions. I believe this to be an unfortunate yet understandable reaction, from their perspective.

The problem, as Hans-Åke Scherp writes in Pedagogiska Magasinet, is that "more control [by the government over schools] isn't solving the problem"{pedmag}.

Keeping microworlds at arm's-length

It seems that fully embracing microworlds is impractical for at least two reasons: firstly, teachers do not have the necessary skills to construct microworlds or to help students explore them, and secondly the microworld model requires the transfer of control from the powerful (politicians) to the powerless (teachers and students).

Luckily, that doesn't mean that microworlds can't be a small part of the existing curriculum, sandwiched between traditional lectures, exercise sessions and exams. Last year (2013), Viktor Rydberg Gymnasium in Stockholm had a mandatory class centred on the Minecraft virtual microworld. Monica Ekman, a teacher at Viktor Rydberg, has described the experiment as a "great success"{local}. This shows that microworlds can be integrated into existing curricula successfully.

The Minecraft model

It's interesting to consider Minecraft as a case study in getting a microworld into the classroom. At the time of the 2013 experiment at Viktor Rydberg,

Minecraft was a microworld with over 40 million users and 17.5 million copies sold. I believe it was this scale that allowed Minecraft to be taken seriously as an educational tool by schools, to the extent that those schools spent considerable lesson-time on it.

That said, as of 2014, no other school has decided to allocate as much resources to Minecraft as Viktor Rydberg, so it looks unlikely that Minecraft will become part of any national curriculum in the near future.

[pos = H, center, botcap]ll This suggests that having 40 million users spending millions of hours in your microworld doesn't guarantee that your microworld will make it into most schools' curricula. This may be a result of the problems of microworlds outlined above. {pedm ag} http://www.lararnasnyheter.se/pedagogiska-magasinet/2014/02/27/mer-kontroll-loser-inte-problemen
{loca l} http://www.thelocal.se/20130109/45514

3.3 Conclusions

Microworlds seem to enable learning in an engaging and effective way, by allowing students to construct knowledge by exploring rich environments. Unfortunately, there does not seem to be the infrastructure in place in educational institutions today to fully embrace microworlds. Teachers do not have the technical skills required to create or maintain virtual microworlds, and politicians arguably have no incentive to allow them to do so.

Teachers at Kungsholmens Gymnasium thought *Narrative Roulette* was a great idea, but could not find lesson-time for me to hold workshops (due to the demands of the politican-set high-school curriculum). I only managed to hold my three workshops when I worked together with a teacher to have them during substitute lessons. And only then when those substitute lessons came up too fast for the substitute teacher to be able to produce a lesson plan for them. It was only in this very particular scenario, when lessons appeared in which the actual curriculum could not be followed, that *Narrative Roulette* was given a chance.

The students loved it{ing:email}. As I argued in Section 10, they also learned a lot.

But only I could maintain the software during the workshops, and even if I improved the interface so that others could run it, the workshops did not provide clear, direct value for the current curriculum. This means that *Narrative Roulette*, in its current form, is only a prototype, not something that can be deployed at scale. And even if the software was improved to the extent that it was easy to use as Minecraft, and even as successful, there is no guarantee that it would be used in schools in more than an ad-hoc basis.

What is needed for microworlds to succeed?

I believe that microworlds cannot succeed until the two problems mentioned above have been solved.

> [1.]Teachers need to have the technical skills to construct and maintain microworlds. There needs to be the political will to have a constructionist national curriculum instead of today's grade-focused curriculum.

When could this happen?

Back in Section 2 we noted that "by the age of 21, the average young person will have spent as much time playing video games as they will have spent in a classroom". This means in the near future, we will have young people entering the teaching profession who have extensive knowledge of playing video games, or exploring microworlds. Hopefully, this means the next generation of teachers will have the skills to maintain virtual microworlds such as Minecraft.

The same applies for the world of politics. In the near future, we will have young people entering parliament who have extensive knowledge of exploring microworlds in the form of video games. Hopefully, this means they will be

better disposed toward constructionist learning, having experienced it first-hand in video games.

However, many people today believe video games to be a waste of time{debate}. Some gamers themselves believe that gaming is a waste of time{reddit}. This suggests that having gamers in parliament may not necessarily lead to more microworlds in the school curriculum.

Neither does having more gamers in teaching and politics mean that microworlds are any easier to *construct*. Constructing virtual microworlds requires expertise in software engineering, as a virtual microworld is a complex software artifact. Today, only developers with industry experience have that expertise.

This is why I believe that it is only when the government employs developers to construct microworlds, and allows teachers to use them, will we see microworlds deployed in education at scale.

Reality check

Today, we are **optimising for administration**. We want work to be quantifiable, because numbers and grades are easiest to file. The more creative freedom a student has, the harder it is to reduce their work to a standardised grade. That's why microworlds are structurally incompatible with the goals of educational authorities today.

According to educational authorities today, the best work is formal, objective, rigorous and in a standardised format.

That means the best test is a multiple-choice test. The best Master's thesis is one with a clear hypothesis and quantifiable results. A longitudinal study of answers to a multiple-choice survey, for example, or an analysis of the performance of different algorithms on the same dataset.

This is not that kind of Master's thesis. The microworlds created for this paper are original and innovative. The discussion of them is opinionated, and driven more by personal experience than academic research.

My work is a result of knowledge I have personally constructed. I learned about microworlds by creating them and evaluating them, in a process that *is itself a microworld*. The three parts of this thesis are themselves the three phases of the microworld loop.

This paper is qualitative rather than quantitative. It argues that learning is a process of iterative, informal experimentation, and is itself the product of iterative, informal experimentation.

That makes this work hard to administrate. Which probably means that it is unacceptable as a Master's thesis, in it's current form.

That doesn't change the fact that I learned a great deal by creating microworlds, trying them on students, and then writing about my experiences. If you've made it this far, and this paper has made you think, then I'm satisfied, regardless of whether this thesis is accepted.

2. _____

{ing:email} Personal correspondence with Eileen Ingulfson, 13 March 2014 {debate} http://www.debate.org/opinions/are-video-games-a-waste-of-time {reddit} http://www.reddit.com/r/changemyview/comments/1ye5w2/i_think_that_playing_video_games_is_a_was

Bibliography

[1] Jane McGonigal,
TED Conversation
http://www.ted.com/conversations/44/we_spend_3_billion_hours_a_wee.html
2011.
Retrieved June 2014

[2] Jane McGonigal,
Reality is Broken: Why Games Make Us Better and How They Can Change the World
Vintage Digital
2011.

[3] Wikipedia,
http://en.wikipedia.org/wiki/Flappy_bird
Retrieved June 2014

[4] Ewan Spence,
http://www.forbes.com/sites/ewanspence/2014/02/18/the-vital-and-depressing-lessons-flappy-bird-can-teach-indie-developers/
Forbes.com
2014. *Retrieved June 2014*

[5] Wikipedia,
http://en.wikipedia.org/wiki/Minecraft
Retrieved June 2014

[6] David Foster Wallace,
Both Flesh And Not
Penguin
2012.