published on 31/12/18 by Stuffed Wombat

Make mechanics & forget the game

People often ask me how I come up with ideas for my prototypes and games.
This text tries to answer that question.

Think in terms of actions and reactions

I am assuming that games are a series of causes and effects.

> *The player presses a button on the keyboard and the character moves.*
> *The character touches a button inside the game and a door opens.*

Certain conventions have arisen in game developement.
We have internalised them without realising it.

> *Arrow keys to move. Tap to jump. Click to shoot. E to interact. Spikes kill you.*
> *Death makes you respawn in the same level, etc etc*

Conventions are not necessarily bad, but they are conventions.
Interesting stuff lies behind them.
To explore these possibilities, it is important to consider that for every action, there is a reaction.

Please stay with me for a few short examples of actions and reactions:

> *If you hold down the right arrow key, your character will move to the right until it runs into a wall.*

Both holding down the arrow key and the player being next to a wall are actions.
They cause a reaction.

> *If you click the left mouse button, your character shoots a bullet towards the cursor.*
> *The bullet gets destroyed when it hits a wall.*

Both the clicking of the mouse button and the bullet hitting a wall are actions.
They cause a reaction.

> *If you click the left mouse button, your character shoots a bullet towards the cursor.*
> *When the bullet hits a wall it gets destroyed and the player starts moving towards the point where the bullet hit the wall.*

Both the clicking of the mouse and the bullet being destroyed are actions. This time they even cause an interesting reaction.

Any action can trigger any reaction.
A single action triggering multiple reactions can create interesting mechanics.

Examine movement

I only make 2D games, which makes this a lot easier.
In 2D space, all movement can be expressed through two numbers:

Angle and speed.

You can simulate complex movement by changing these numbers over time:

Increasing speed is called accerlaration.
Rotating the angle of movement is called turning.
Inverting the angle of movement while decreasing speed simulates bouncing off of something.

This might sound like super basic stuff, but thinking about movement in these terms has enabled me to look at games as a simple collection of movement sets.

Look for Loops

Actions, reactions and movement are not taking place in a vacuum.
In conventional games, players expect to reach a goal and finish the game at some point.
To do that, they perform a set of actions over and over again.
These repeated actions and their reactions can be called a core gameplay loop.

For example, here is the core loop of golf:
apply force to a non moving opject
object moves
object comes to rest
repeat until object is in the goal

This is pretty basic stuff, and again it enables us to see games in a new light.
If one wanted to create a new take on golf, each of the steps of the core loop offers concrete possibilities for deviation and innovation.

Here are some examples:
players can apply force to the object while it is moving
the object does not move, the goal moves
the object never comes to rest
the goal is to not let the object reach the goal for as long as possible

Being able to recognise gameplay loops enables us to see games as a collection of actions and reactions, each of which can be subverted to create a new core loop.

Prototype a lot

Prototyping is a skill that can be learned.
The only way to learn it, is to prototype.

What has been incredibly helpful to me in this process is the fact that I can not code well.
Using visual scripting and an engine that has a lot of pre-build movement sets enables me to iterate pretty quickly over ideas.

When coming up with ideas by means of examing actions, reactions and movement, the results are often absolute garbage, so it pays off to be quick in testing them.
I also try my hardest to keep my projects very small, again increasing the speed at which I am able to iterate.

You can use your awareness of actions, reactions, movement and loops to come up with an endless stream of mechanics for conventional games.

Anyways, here is the core loop of creative work:

attmept to do something
fail
despair
repeat

Forget the game

Creating innovative mechanics is fun, exciting and makes you feel very smart indeed.
It is also pretty addicting.

It is also only a small part of game developement.
Much more challenging problems arise much later.
There is no time to talk about that here, but please believe me:

Finishing a mediocre game is better than to never finish a hundred great games

I used to think that I had to come up with extremely clever mechanics to make a good game.
Downwell, VVVVVV, Braid, all the games I look up to have incredibly unique and smart mechanics.

I thought that I had to have the same if I wanted to suceed.and after prototyping for months, a curious thing happened:

I got so used to making mechanics that I forgot how to make a game.

Being able to prototype interesting mechanics quickly is pretty neat, but it does not turn you into a good game developer.

To become a good game developer, you need to develop games.