# [Devlog] In Eggcelent Condition

Welcome to my devlog for the game “In Eggcelent Condition”!

At the start of December, the “New Horizons” game jam came on my radar. Even though I’d already done *two* huge game jams in November, and wasn’t really looking to do another, the theme just resonated with me: make a game that’s *positive* and about *new year’s resolutions*.

This is the game I made for it. Below you can read all about how it came to be, why I made certain decisions, and hopefully it’s both interesting, fun and educational to read :)

## What’s the idea?

Well, yeah, that was a real problem. I was quite busy, so I wasn’t able to come up with some semblance of an idea before the jam started. I waited one or two days, but nothing really came to mind.

So I started brainstorming. Quickly I realized I wanted to do something with “eggs”: they signal a “new beginning”, they could be a metaphor for your wishes/desires/dreams coming true, it felt fitting.

But it felt too “thinky”. I already did some puzzle games recently, and didn’t want to add another one about collecting/hatching eggs.

I kept brainstorming. Then I realized that I didn’t have much time left and should do something simple. Something silly. Something “childish”, perhaps.

That’s when I wrote down the second idea: *you’re just trying to keep something in the air*. (When it touches the ground, you lose.)

A few minutes later, I realized I could combine these ideas:

* You’re trying to deliver eggs from A to B
* You must keep them in the air, because they’ll break if they touch the ground.

And that became the initial idea!

## Making it unique

Delivering stuff, or keeping it from breaking/falling/dying, isn’t new to games. So I wanted a twist on the concept.

Eggs have quite a unique shape, making them really hard to catch, bouncing all over the place. This indicated (to me) that this should be a *physics-based game*, with proper eggs that realistically bounce around.

Instead of just *grabbing an egg* and moving it … **the player has no hands!**

In my head, I envisioned this concept of:

* Players have *huge* rectangular heads
* Use them to bounce the egg up, keeping it from hitting the ground.
* Until it falls into some *egg basket* ( = it’s delivered)

So I made that. And then I realized: this is *way too hard*.

I tried the following things:

* Make eggs bigger => good for clarity, but there’s a limit here, after which the screen just becomes an ugly mess :p
* Give eggs slower gravity => fixes the issue, but makes the game feel slow and boring, and doesn’t match the player speed
* Make players way faster => again, fixes the issue, but there’s a limit after which players are just zooming around the field and you don’t know what’s going on
* Make their head bigger => yes, it’s easier to save eggs, but only if it’s *so big* that the proportions look *very weird*
* Give them something else, like a racket or frying pan, to save eggs => actually a great idea, but not as the *core concept* of the game, more like a powerup.

Then I realized I was thinking the wrong way.

Why does it *have* to be the player’s head? Why can’t it be a … **huge sombrero?**

I gave players a huge disc on their head, and the game instantly became more fun and playable. (The 3D model comes later, for now it’s just grey cylinders and cubes.)

But it wasn’t enough. Yes, you can *save* an egg from falling … but you can’t *steer* it. The eggs keep bouncing however they like, and you just have to hope they land in the right spot :p

Luckily, the “sombrero” theme gave the answer. If I add a round “ball” or “hill” in the center of the disc, you can use that to send eggs whichever way you like.

And that’s how we got here:

* Players are just a huge sombrero, mostly flat, but with a ball in the middle.
* Eggs do nothing special and just follow physics laws.
* Your job is to keep them off the ground and make them land inside the egg basket.

TO DO: IMAGE

## Creating Eggs

I knew I wanted many different *types* of eggs with *patterns* on them. (Like chocolate eggs you eat during Easter.)

The *inefficient* way to do this, would be to duplicate each egg, assign a new material, and repeat the process 20 times.

The *efficient* way to do it, is this:

* Grab the default cube in Blender
* Scale down the top face
* Give it a *subdivision surface* modifier => now it looks like an egg!
* However, because it’s still just a cube, the UV-map is easy to work with!
* I created a texture with the pattern for each egg, in a neat grid layout.
* In the game, I simply *offset* the UV coordinates to select which pattern I want.

TO DO: Image

This way, we only need *one model* and *one material*, yet we can get infinite visually appealing eggs!

## Destroying Eggs

I also knew I wanted a proper *break* animation when eggs hit the ground.

Again, we can do this quite easily in Blender:

* Activate the “Cell Fracture” add-on.
* Duplicate the original egg (with the right UV coordinates)
* While selected, SHIFT+R, search for the fracture add-on, and click it.
* Apply it. (You can play with the settings, but I saw no need.)

Now you have an egg divided into several chunks. The original UV coordinates are maintained, so it still looks the same.

All the *extra faces* that have been created needed to be mapped again, so I just gave them all a solid color. (The same as the egg itself.)

I imported this to Godot, generated a collision shape for each mesh (using Godots automatic tools for that), and put mesh + shape inside a RigidBody.

Now I had an egg … but scattered among 8 separate bodies! Which is what I wanted!

When an egg hits the ground, we instantiate the “BrokenEgg” scene, with the same position/rotation as the original egg, and the same pattern.

For an even nicer effect, we apply a random force to all the parts, to really scatter it.

TO DO: Image/GIF

Cool! I’ll have to check in with performance as the game grows, but it seems fine for now.

## Level Layout

I struggled with the layout of the levels ( = world/arena you play in).

Here’s the thing:

* Eggs shouldn’t fly off-screen all the time. They should be contained within the level.
* But adding invisible walls is annoying, as you don’t know exactly *when* an egg will bounce, or *where* players can move.
* But adding visible walls, well, *obstructs what you’re doing* (and looks ugly).

After some thinking, I realized I should integrate the walls with the level design.

What do I mean with that? I should create an environment *around* the level, that looks natural, but has the extra purpose of keeping you locked in. So, not an invisible wall, but a series of blocks and trees and other things that don’t look out of place.

In fact, I found this crucial enough to work on right now. (While testing, it became *really annoying really quickly* that eggs would bounce against invisible walls and I didn’t know where I could walk.)

Having never done something like this before (in fact, this is one of my first 3D games), I just gave it my best shot.

The good news? Yes, it works way better! You know exactly where the level stops, eggs bounce back predictably. (The invisible wall is still present, just farther away, to catch the eggs that *really* go flying.)

The bad news? It doesn’t really *look* great … yet. This might be because of:

* Colors
* Lack of actual environment (trees, rocks, etc.)
* Scale (it feels like the tiles should be two times bigger, to match better with the players and eggs.)

But those are worries for later, let’s continue with more essential stuff!

**Remark:** For those wondering how I did this. In Blender, I just gave the cube some grass on top and dirt pattern on the side. Then I duplicated the cube a bunch of times and made adjustments: a slope, a steeper slope, etc. Exported it, turned into a MeshLibrary in Godot (there are tutorials for that online), and used that in a GridMap to draw a semi-random environment.

## Obstacles

Creating this level *did* give me an extra idea though. Why give the players a *flat, blank, rectangular* playing field?

Isn’t it way more interesting if the field itself *also* had certain obstacles, variations, items that you might use?

There could be a *slope* that bounces eggs in a less predictable way. There could be a *pillow* you can shove around, which can safely catch eggs of course. There might just be small rocks and plants that can get in your way when walking around.

It would bring more life to the arena, both in *visuals* and *gameplay*.

So far, I’ve just written down a list of ideas, but haven’t implemented anything yet. I don’t think that’s the priority right now.

## The simplification step

If you’ve read other devlogs of mine, you know this step always arrives: we have a basic concept, we have many possibilities/ideas that could be fun … but we can’t throw them *all* into the game and expect people to understand it.

Let’s simplify instead.

### Step 1: powerups

One thing this game surely needs are *powerups* (or *something* to change physics properties and play with them). But I don’t just want to randomly place some floating icons in the level.

(Why? It would require players to learn even *more* icons. It would be random, players have no say in what they will get. It would feel tacked-on.)

Instead, let’s **make the eggs the powerups!**

Here’s the idea:

* There are different types of eggs. (Indicated by their color/pattern.)
* If an egg *breaks*, it reveals its powerup. Now it becomes a floating icon of that egg, and you can grab it.
* If an egg *is delivered*, well, nothing more happens.

Why do I think this is a good idea?

* Breaking things *still* gives you a way forward. (Helps with the theme of “positivity” and “new beginnings”)
* There’s a strategic choice here: do you want to deliver an egg, or break it for its powerup?
* There’s only *one* list of things in this game, and that’s the list of eggs that can appear. Their icon/pattern/color stay constant. (In fact, I’m thinking about teaching nothing up-front, and just showing the eggs currently in the game in the top-right corner.)

### Step 2: buttons

Over the years, I’ve learned that every extra button is one too many. (At least, if you’re aiming for accessibly, family-friendly, coop games like I do.)

So I want this game to be playable *just* by moving around (and thus learning the arrow keys/joystick on controller).

However, actions like *jumping* or *dashing* seem quite essential to a game like this. So I still want to add them and make them easy to execute.

How do we do that? Well, we **make actions (that need a button press) powerups as well!**

You can only jump once you’ve grabbed the *jump* powerup. When you have it, the icon appears above your head, *plus* the button you need to press.

This means no explanation up-front, and a constant reminder in-game if you can do something.

I’ll have to see if this isn’t just annoying (because you can’t jump if you don’t have the powerup), but it’s a good idea to keep for now.

### Step 3: summarizing the game

Let’s do a check for simplicity by summarizing the game.

* Objective? Deliver X eggs.
* How? Move around and hit them, so they *don’t* hit the ground, but *do* land in a basket.
* They hit the ground? They break and their powerup can be grabbed.

Any other input is shown when needed. What eggs do is shown in the corner (for all eggs chosen for *that* particular game.)

I think that’s quite simple enough.

## The physics-control step

And as with all *physics-based* games, there comes a time where you must take some control back into your own hands and restrict the physics system from doing stupid stuff.

(Completely realistic physics are, in most games, not fun or interesting at all.)

These are tiny things that made a huge difference for this game:

* Whenever an egg hits something, its bounce is *increased*. (By quite a large factor.)
* Not just that, the *upward* component is also increased. (So eggs will mostly bounce up into the sky, giving you more time and a clearer view of where they will land.)
* Their maximum speed is limited, just in case something odd happens. (Don’t want eggs flying off the field with the speed of light!)
* On egg baskets, all bounce code is *turned off*. (Otherwise, eggs could easily bounce *out* of it again, or hit the rim and fly all the way across the field.)
* Players have a way higher gravity than eggs (or any other elements).
* There’s no realistic damping. Player movement is dampened/stopped quite harshly using my own code, otherwise you get a “slippery slidy” player controller that just makes this game impossible to play.

There’ll probably be many more things as we go. These are just examples of what you can do to make physics-based games a bit more controlled, instead of random and frustrating.

## Creating the content

And now’s the moment where I just buckle down and *implement all those ideas I have*. Throw everything against the wall and see what sticks.

This means:

* Drawing a unique (recognizable) egg pattern
* Assigning some powerup or specialty to it
* Revealing it when the egg breaks + making it “pickupable”
* Creating the functionality in-game for it

Repeat a bunch of times, until I run out of ideas. (Also, preferably, the egg pattern would be a *hint* towards its functionality. The jump egg gets an arrow pointing up, for example.)

## Casual Testing

While doing this, I constantly test the game of course, which yielded the following observations.

### Levels & Environment

About the level layout and environment …

* If we’re going to spend so much time creating an environment *around* the arena … why not make the whole arena a unique environment?
* The game could have a handful of “levels”: one in the desert, one in the forest, one in the city. Each would use that same GridMap as before *for the whole thing*, and then add new models on top.
* This way, I can also get more creative with my boundaries. Instead of “invisible walls”, the boundaries of the level could just be *buildings*, or a *tree line*.

(Also, I updated the core tileset to be *bigger* and have more saturated *colors*, and it already makes it look way better.)

### Core Gameplay

* Getting the egg inside the basket is *really hard*. (No matter how big I make the basket and your sombrero for controlling the eggs.)
* The basket is also quite *high* right now, which means it’s easy to misjudge, ending with eggs just hitting the side and breaking.
* When I make the sombrero model, I should also make sure it has the same *hitbox* as an actual sombrero. Why?
  + The ends curl upward a bit on sombreros, which would help keep the eggs on your hat.
  + The bulb in the center doesn’t have perfectly vertical sides, they are more gradual, which *also* helps with controlling egg bounces.
* Often, it feels like I’d want a way to deliver the egg *in the air* (like a hoop it should go through)
* And that it might be more fun if the level *wraps around*: both eggs and players simply teleport to the opposite end when they reach the border.

The last thing I was constantly doubting was: “should the sombrero be *high off the ground* (so that it stays above baskets and catches eggs earlier) or *very low to the ground?*”

In the end, I took the second option. Otherwise, eggs that were already low to the ground simply *could never be saved*. Additionally, the higher your sombrero, the *harder* it was to gauge where it actually was and how it would intercept the egg.

So, your character is now just this *tiny* cube, with the sombrero directly on top of that :p

### Conclusions

**#1:** Yes, with a bigger sombrero, with a better hitbox, controlling eggs is quite doable now.

**#2:** Level wrapping (things that go off at one side, reappear at the other) was a great solution … until I added the sombrero as a separate physics object. Now I’d need to teleport *two bodies,* which are *linked together* (with a “joint”) … yeah, that’s asking for trouble.

I also planned a powerup where you could *throw* your sombrero, which made the level wrapping definitely a bad idea.

I’ve given up on the idea of “randomly generated arenas” (so with the egg baskets, cannons, etcetera at random locations).

**#3:** Instead, I will create ~5 hand-made arenas, where I manually place everything and it’s fixed. This way I can find creative, fitting ways to add bounds to the level *and* add special rules and variety to each arena.

**#4:** The sombrero is attached to your head with a loose joint. This means it wobbles when you move (which is just fun), but it *also* means that it realistically adds extra power to eggs!

(When they land on the sombrero, it dips down a bit, then shoots up, adding extra force like a spring. The more an egg lands on the *outside*, the more force is added – again, because of the real-world concept of *leverage*.)

So I’m definitely keeping that and making it a core part of the game.

**#5:** The sombrero *only* interacts with eggs. It can go through terrain, egg baskets, other players, etcetera, without issue or collision.

Why? Because you otherwise couldn’t get anywhere (at least without bumping into stuff and being stopped and having to jump your way through). The game flows *so much better* when you can just position your sombrero anywhere you want.

**#6:** Some of my powerups change physics on nearby eggs. The most efficient way to do this (by far) is to use Godot’s Areas for that and turn on “space override” (with the new physics properties I want). Holding the button? Turn override on. Released it? Turn it off again.

However, it wasn’t working. And I couldn’t for the life of me figure out why.

Then I stumbled upon this comment by someone in the Q&A forums: *areas are only updated when they are moved*. From my own tests, I’d concluded that *areas are NOT updated when you change their space override (toggle on/off).*

So I had to change it to the following: the area override is *always on*. However, the *size* of the area is set to 0 when it’s not active. Additionally, whenever something changes (button press/release), I move the area by a random (tiny) amount to force it to update.

A bit annoying that there’s no clear documentation on this (or an easy method to mark an area as “dirty”), but at least I figured it out.

## Arenas

Okay, so, what do we have now?

* A wide variety of eggs can appear
* You can balance and push them quite well with your sombrero
* They can be delivered into a basket or smashed to get some powerup.

This means that arenas are the final remaining “essential thing”. (After that, it’s time to start looking at polish, such as sound effects, prettier graphics, minor fixes, etcetera.)

As usual with these games, I wrote a list of “what each arena *needs* to be good”:

* Unique color scheme and decorations. (Arenas should *look* and *feel* distinct.)
* A natural way to block off the edges.
* Either a special (general) rule …
* … or a special way of delivering the eggs. (Just dropping identical egg baskets in every environment is not fitting nor interesting.)

And then I brainstormed until I had an answer to all of these questions. It doesn’t necessarily need to be a *good* answer or the *final* answer, just something to try for now.

It yielded some interesting ideas for arenas … just not sure if I have the time (or 3D modeling skill) to pull it off :p

For example, I’m thinking of adding “Easter Island” (you know, with the big stone faces dumped into the ground) and “Cloudy Babies” (set in the clouds, storks holding towels/cloth, your egg must land in there). Thematically speaking, awesome. Attainable? Not sure.

But the only way to find out is to keep working and see where we end up.

## Quality of Life

There are always many, many ways to make the game clearer and easier. And they usually only pop up if you actually *test* and *play* the game for quite some time (in different situations).

This is what I found:

* We need an indicator on the ground that shows where an egg is going to fall.
* Similarly, an indicator when an egg is off-screen.
* We need outlines around eggs.
* A ceiling is also necessary, otherwise eggs can fly *super high* and you never know when they crash back to earth.
* An interactive menu would be great: one that teaches you how to move around, because you *need to move to a spot to start that level*. (It’s more work than a traditional menu, but it’s miles better. It means the game starts right when you launch it, not only after clicking some menus. It teaches the game, without taking extra time, or it feeling like a tutorial.)

## More Physics Annoyances

Today I implemented my special sombrero powerups: throw it like a frisbee, put it at an angle, etcetera.

To do so, I need to

* Destroy the joint that attaches the sombrero to the player
* Position and rotate it correctly
* Add the new joint (if needed)

Oh boy, this was a pain. The basic version was working within a minute: create the specific joints (that attach sombrero + player) at runtime, allow destroying the old one.

However … this only worked when the player was at the perfect starting position (0,0,0). And once a certain joint had been chosen, switching to a new one later (by e.g. grabbing that powerup) created all sorts of weird glitches.

**Remark:** I was also being stupid, as I forgot to write the line of code to actually *remove* the old joint, instead of just *checking if there was one*. That’s why I was confused for 20 minute as to why joints didn’t do the very simple thing they were supposed to do: there were 2 or 3 joints acting on the body simultaneously.

In the end, it came down to what it always comes down to with physics: they are updated in a *separate step*, which means it *might take a frame (or two) before the physics are actually correct and lined up*.

So this is what I ended up doing:

* Grabbed powerup? Remove the old joint, *plan* the next one => we wait a few frames
* Check the *planned* joint
* First, we position the sombrero correctly for this specific joint. (For example, the regular one is just at the top of our head, with no rotation) => we wait a few frames
* Now create the joint and set the right parameters

Removing a joint takes a frame to update. Forcefully setting a physics body position takes a frame to update. That’s why there need to be pauses between all these things.

I’m sure there is *some* way to just say “body.force\_physics\_update()” to get it immediately, but I couldn’t find it after searching for hours. (If it’s not there, it *should really be added*. At the same time, there’s probably a *reason* it’s not implemented then …)

Of course, if the player *moves* during those pauses … the joint is added at a slightly wrong position! So, while the joint is “settling” (that’s what I called it in the code), the player can’t move or rotate.

This means you’re locked for a few frames, but it’s short enough to not be noticeable.

(The alternative is *hard-resetting* the sombrero to the correct location every few frames. But that’s very resource intensive and opens up many more troubles, so let’s not do that.)

But, after all of this, we now have a bunch of fun powerups for throwing, slanting, repositioning your sombrero. (And they all have some wiggle room, so they look bouncy and react realistically to the eggs.)

## Nearing a playable game

### Menus

I decided to make the menu interactive. In fact, I decided to simply make that “one of the arenas”.

(So it’s built from the same GridMap, follows the same physics rules, etcetera. Only, instead of playing the game, you can log in walk to a spot to load that level.)

This means that I just need to *make that arena* and add the functionality to complete the “menus” part of the game.

(A specific button opens up *settings*, but that’s basic stuff like “Volume” and “Fullscreen”. A pause menu in-game would be nice too, but that also won’t be much work.)

I’ll start now, but obviously need to wait until I have all the arenas to finish it, because at the moment I only have basic *grass* tiles.

### Solo Mode

Using custom-made arenas, means I can’t really scale them based on player count. That means I need to scale them *to allow 4 players at most* … which means it feels really big and empty when you’re playing single player. (And that makes it near impossible to play.)

That’s why I decided to give you **two sombreros!**

On keyboard, you simply control the first with arrow keys, and the second with WASD.

On controller, you get a button to *switch* between them. (Might also allow moving the other with the right joystick, but not sure how easy that is to execute for players.)

### Tutorial?

I decided to make the first arena a “training ground”, based on my experience creating multiplayer games the past few years. Yes, it feels like a “boring” start, but I know it will be the “perfect” start for new players.

This map has some rules and mechanics omitted:

* Eggs *auto deliver*. If you can keep them in the air ( = so they don’t break) for 10 seconds, they are delivered.
* When eggs are broken, they *don’t spawn powerups*. (In other words: special eggs and powerups are disabled.)

I’ll make sure the maps are ordered and numbered so players will start with the training ground.

The next map will then add some rules:

* Eggs must be delivered into baskets.
* Powerups spawn, *but* only a few simple egg types are enabled for the moment.

The map after that will add *more* unique eggs, and so on, increasing difficulty as we go.

How are these things taught? Although relatively effective, I’ve grown to despise hitting players with a “full screen image tutorial” at the start of a new level/arena. It breaks the flow. They won’t read it all. (Probably, someone will skip it and say “heck, let’s just play!”)

Instead, each arena has a small image that appears when you come close. It tells you:

* The name
* A thumbnail of what it (roughly) looks like
* Any special rules or newly added mechanics to watch out for.

That’s all the tutorial needed, and that’s all you get.

TO DO: Image of the arena tutorials?

### Some more help & variety

Although it’s now quite doable to control the eggs and keep them from breaking, this also means some of the *variety* and *challenge* is gone.

I realized I often *really* wanted to jump to an egg. Or dash towards it. Things you only get when that powerup appears.

But … we can also give this functionality to you *through level design*!

If I simply place “trampolines” in the levels, you can use those for some jumping. If I place “dash squares” in the levels that boost you in that direction, you get the same idea.

This does mean, though, that levels get a bit fuller and shouldn’t be so flat.

The current method for checking “does an egg hit the ground?” was, well, just a rectangle on the floor that checks if the egg hits it. That’s not possible anymore if the ground is all bumpy and uneven.

Instead, I’ll change it to this:

* There’s a *separate* GridMap called “Floor”.
* Only eggs that hit the Floor are broken. (The other GridMap is used for the edges, environment, decorations, etcetera. No fun if those break eggs too.)
* Additionally, they only break if the collision normal is pointing *upward.* (In other words, they fell down on it, instead of hitting e.g. the *side* of a tile, or a *ceiling* of something.)

It’s not precise, but then again, physics games are never precise. There will be eggs hitting the floor that *should* break, but they don’t, and vice versa. But if that only happens 1% of the time, we’re fine.