# TO Do (Sheepe)

## Coding improvements

* Convert “Room” object into an actual scene, with nodes as modules (main, outline/edges, terrain, painting, etc.)

## Tutorial

* Allow placing “tutorial images” anytime during a level, and then activating the rule we introduced.
* Create a campaign overview screen + make “next level” button actually go to next level.

## Gameplay

**Step 0:**

* Start players *bigger* => scale the predefined shapes at runtime
* Severely reduce max speed / acceleration in air

**Step 1:** I should use the fact that players can be **any shape** and that this can change more.

* Idea: Your **size** (or “mass”) plays a huge role. (You’re faster when you’re bigger?)
* Idea: Your **number of parts** plays a huge role. (Gates you can only pass through if you have *fewer than* or *more than* the indicated number of parts?)
* Idea: Maybe there are specific “gates” with weirdly shaped gaps. (Like that TV programme where you had to stand in a certain pose while a shape came towards you.) You will have to find one that *you* fit through.
* Idea: There are sections/powerups that *reset* you to a specific (predefined) shape. **Or just your original shape.**

**Step 2:** I should use the fact that you’re **rolling** and the keys are **simple** much more.

* For the first level, *only explain the roll right key?*
* Create a door which you must open by rolling against it?
* Create powerups that are an **edge** (or attach to an edge) which you get by *rolling* over them?
* **The more you keep contact with the ground ( = not in air), the better?**
  + Like a snowball effect, you *grow* by rolling over the floor?
  + Or, you *become more round* as you roll, and *become more malformed* as you stay in the air?
  + **TERRAIN IDEA:** By default, the rule is “rolling makes you round”. But in this terrain, “rolling makes you bigger” (and air makes you smaller).

## Bugs

**BUG:** Last body of a player can be deleted right now. Prevent this. (Keep a counter somewhere with “number of bodies by player X”. Probably player manager.)

**BUG:** Sometimes it counts collecting a coin as collecting *two coins*. (Sometimes even three???)

**BUG:** What if the number of total players bodies exceeds 20? Then the “fog of war” shader can’t handle that. Two solutions:

* Either never allow this. (Don’t slice if it would get the total body count too high. Or remove to make space.)
* Or *sort* bodies based on their room, and just *ignore* lights on the ones furthest behind.
* Third solution => don’t give PLAYERS lights, add them to ROOMs? (Or give players a light when there is *none nearby.* Only add lights to large rooms.)

## Movement Improvements

**Step 1:** Add something to “help” flatter shapes roll. Or should I keep it as-is and just expect players to be strategic and *stop rolling* if their shape is bad?

* There should be clear situations in which a *round shape* is good, and other clear situations in which a *flat shape* is good.
* **GREAT IDEA:** There are buttons on which you must *stand for a few seconds* to activate them. If you roll well … it’s hard to stay on it. If you’re flat, it’s very easy.
* **IDEA/THOUGHT:** Flat shapes are better when glueing parts back together, aren’t they? It will fit more nicely.

## Map Improvements

Make the **extra tiles algorithm** work:

* Only update bitmask *once*, not after *every single function* called in slope\_painter
* It still somehow yields blocked-off rooms sometimes? => is this fixed??

**BIG ISSUE:**

* **Create outline** algorithm doesn’t work anymore => it creates it around the *real size*, and it can’t tell the connections to other rooms anymore
* **Fill room** algorithm doesn’t work anymore => because the neighbors are *also from the same room* (guaranteed, as we only check inside, and rooms are one bigger than they appear)

**BIG ISSUE:** It often places a teleporter when *it really doesn’t have to* => still the case?

## Slicing improvements

**Step 1:** Be way more precise with intersect\_shape => create a *rectangle*, the length of the line segment, narrow width, rotated + positioned around angle + avg.

**Step 2:** When sliced, fling the bodies apart. (Looks better/more fitting, but also prevents lots of double slicing right after each other.)

**Step 2:** The clinging force + jump force should be *proportional* to player size. (Otherwise small pieces get stuck, and large pieces cannot cling.)

# Done

## Basic Bodies

**Step 1:** Generate a random polygon

* <https://stackoverflow.com/questions/8997099/algorithm-to-generate-random-2d-polygon> => basically, create a circle, but allow each point to vary in radius/angle
* <https://stackoverflow.com/questions/59287928/algorithm-to-create-a-polygon-from-points> => draw a point cloud first, order by angle, then draw through it

**Step 2:** Calculate its centroid. Place a smiley face there. Then center the polygon around it.

**Step 3:** Turn it into a physics body + draw it each frame.

**Step 4:** When given input, roll in a certain direction. (Check if this actually works for movement.)

## Body slicing

**Step 1:** Write the slicing algorithm I scribbled on paper.

* <https://stackoverflow.com/questions/563198/how-do-you-detect-where-two-line-segments-intersect> => detect intersection point of two lines
* The rest of the algorithm is just:
  + Loop through shape.
  + Detect first intersection point. Add it to the shape. (Between the start/end vertices of the edge it intersects.)
  + Continue until second intersection point. Add it to the shape.
  + Now *extract* the part between the two points: shape 2. *Remove* the part you extracted from the original shape: shape 1.
  + Now recreate the *bodies* + *draw/move scripts* for each.

**Step 2:** Allow testing by drawing with the mouse. (Or clicking twice. Or pressing a key and testing a predefined line.)

**Step 3:** If successful, allow applying dynamically.

# Discarded

The old idea with “placing precreated rooms”

## Rooms & Routes

**Issue 1:** How do we allow *rotating* rooms?

* Translate everything to anchor center
* Rotate the thing
* Translate everything back => DOESN’T WORK, because the “position” property is still local, so translating back would just *follow the new orientation*
* Now recalculate opening values

**Issue 2:** What if a single side has *multiple* openings?

* We should be able to match any of them
* But *not* necessarily close the others when filling gaps

**Issue 3:** Now we have ugly *double walls* between rooms.