# TO Do (Sheepe)

## Coding improvements

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

## Performance things

* Only do “update\_bitmask” once, when *everything* is done.
* Remove *modules* from player bodies that don’t need them. (For example, *tutorial* module only works on first body of player.)
* We’re looping through full rectangles awfully many times. Instead:
  + Combine these loops to calculate multiple things at once.
  + Or save all positions *in a list* and just iterate that list. (Instead of positions, we might also save a reference to the cell itself. Is even faster access.)

## 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

**THIS IS THE IDEA!** There are no powerups floating in the air. All special items, all powerups, all interactions *are done by touching/rolling over a certain piece of terrain*.

* This way, I only need to place sprites **on top of** existing tiles. (And match their position + perhaps rotation)
* And the space itself stays clean for jumping and bouncing around

**POWERUPS:** Get them by rolling over the tile on which they are placed.

**IDEA:** A block (inside rooms) you can break by *hitting it with speed?* (And breaking might get you something?)

**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?

**IMPROVEMENT:** Add a delay when switching wolf. (Alternatively, add a delay *in general* when determining trailing and leading player.

* Bonus: the wolf moves *faster*
* Restriction: the wolf slowly shrinks => not only is this a penalty, it also ensures that the last player *will* be able to get through obstacles.
* Easier => just make the wolf *split someone in half*. No need to calculate the exact collision position, as it often isn’t visible anyway, or leads to no-slices.

## Bugs

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

## Map Improvements

**FILL ROOM Algorithm:** Add a variation where we’re allowed to place tiles *against the walls*, but *not in the center*. (By default, we only place away from walls, in the center.)

**TERRAIN IDEA:** By default, the rule is “rolling makes you round”. But in this terrain, “rolling makes you bigger” (and air makes you smaller).

**THING 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.

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

**BIG ISSUE:** At the start, there should not be a wolf. (Otherwise mayhem already ensues, before players can even react.) Nor should there be a terrain or any other fancy stuff. Just create a general: “params.disallow\_everything” and activate on start room?

## Slicing improvements

**BUG (?):** When glueing objects back together, it sometimes crashes? Because the room.players\_inside array has a few null entries. (Which would be caused by a body being killed but NOT removing itself from that array first.)

**BIG ISSUE:** It can’t really calculate the area/size of a body now. (It calculates them based on orthogonal x,y … but most shapes are rotated in some way or another.)

* **Solution:** with the new area algorithm (shoelace), this seems *mostly* fixed anyway.
* **Solution:** rotate the shape once (45 degrees), calculate again

# 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.