# TO Do (Sheepe)

## Gameplay

**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.

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

## Gameplay (Essentials)

**Step 1:** Once *everyone* has finished, show an overview of the ranks + times on screen.

## Annoying situations

**Situation #0:** When backtracking, I have to allow more free placement. (Adjaceny/overlapping old rooms.) But … this leads to unclear routes, as there’s *no way* to know where you should be to count as “being in one of the front rooms”

Maybe I could add *edges* between the new rooms and the old ones? (These can separate rooms *without requiring much space*.

**IDEA:**

* Give the “previous room” to each rectangle
* Place edges along outline, but *not* when the previous room is on the other side
* Once this works, I can “scale it back” to only use when needed.

**Situation #0:** Don’t allow the **very last body** of a player to be removed. If this is about to happen, simply don’t allow their body to be *sliced* in the first place?

**Situation #1:** Two triangle slopes right after each other => often there’s no space to squeeze between there.

* If this happens … place a “cutter” before it? (So players have a *chance* to reduce their size?)

**Situation #2:** Growing a teleporter, and just plopping it down (+ erasing all around it) … looks ugly. Additionally, sometimes it happens *way too often* and *way too quickly.* How to ensure paths have a bigger chance of survival?

* Create *edges* on the outline. But this time, keep them open when there is *any* free tile on the other side. (Not just in the last room, but all rooms.)

## Movement Improvements

**Step 1:** Cap influence of air resistance. (Perhaps add a force that slows you down *when in the air*, towards a maximum velocity. So you can still go *really fast* if you gain speed, but moving in the air will always be limited.)

**Step 2:** Might need some extra *safeguards* to ensure you can always move somewhere successfully.

* Stronger jumping?
* Getting bitten by the wolf behind you, somehow *resets* you or gets you *unstuck*.
* Should I add a *counter*, and if you stood still for 10+ seconds, you “explode” or “get a boost”?

**Step 3:** 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.

## 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:** 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.