# TO Do (Sheepe)

## Essentials

* Locks
* Powerups
* (Solo mode)
* (Menus and all)
* Soundtrack + sound effects

## 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.
* The first time a certain lock appears, place a tutorial in the background for it

## Coins

Only show coin interface when *collecting one* or *whenever it’s relevant*. (We’re inside a lock that does something with it, we’re near a powerup that does something with it.)

Bigger question: what exactly are coins good for? (It feels a bit tacked on at this point.)

* Coins protect you from the wolf. (They just take a coin, instead of biting you.)
* Some locks require them *or* make your life easier with coins.
* Some terrains use them.
* Some elements should also require payment? (Make sure it doesn’t become overly complicated!)

## Bugs

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

**BUG:** When backtracking, it doesn’t properly update the previous room. (It just adds another special item according to the *current* state.)

## 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.)

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

* Here’s an idea. When we can’t place a room, just ignore it. Only when the buffer between leading player and end has become <= 2, do we transform the last room into a teleporter.
* (Absolutely no need to do it earlier. And this gives the map time to clear out and maybe find a solution later.)

## 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

## Polishing

* Whenever I do something to a jump normal, show a tiny line for that. (Similar to debug, but prettier.)
* Add “bouncy” tween to the *shaper* node when jumping or hitting stuff.
* Give an indication when someone is holding *both buttons* => perhaps show a different color or overlay when *floating*. (Add wings at the side?)
* Give feedback
  + Especially when getting a time penalty or getting/paying coins.
  + But also enable option to give *textual feedback* when someone first enters a terrain, which gives a hint *how* this terrain works.
* Add different control scheme for controllers: joystick to roll left/right, any button to jump/float.
  + (Make this default? Or can players configure it themselves?)

# Done

## Annoyances

**ANNOYANCE:** When you jump with your head against the ceiling, your *rotating* movement actually pushes you in the wrong direction. Which is just … annoying? (Yes, you can learn it, and use it for stuff, but … not great.)

* Solution #0: Make ceilings frictionless => can’t do it, as they’re part of the tilemap, which has *one* physics material.
* Solution #1: Always cling to ceilings => possible (check if cling vector is opposite to gravity vector)
* Solution #2: Make jumping less powerful
* Solution #3: *Hold* both buttons to *float* or *steady yourself*. (So when you hold both, your Y-velocity becomes 0. But your X-velocity continues.)

## 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.