# Rolling in the sheepe

## General

Each player is a **random shape**. You can only **roll** through a world full of obstacles.

The map flows to the left, building itself as you go along. (There might be split paths, ways to go down, direction reversals. But keep it simple at first.)

**IDEA:** There are obstacles that *glue* you to them.

**IDEA:** There are obstacles that simply hurt you (so you slow down/reset). But also ones that *split* your shape, like a spike shooting through you.

**IDEA:** When this happens, you simply roll *both of them*. They can be recombined later. But you only win if *all of you* crosses the finish.

**IDEA:** Alaser might also just shoot through at certain moments, slicing *everyone* it hits.

**IDEA:** a way to “lock in sections”. So, players that are further ahead must *wait* for the rest to catch up. (But they might get *bonuses* for arriving there earlier. Or there are things to do while “waiting”. Maybe it’s a *minigame* they need to solve to open the door.)

Why? This ensures players don’t go too far apart, allowing the camera to keep all in frame.

Why? It’s the “Mario kart” way of keeping things competitive.

## Theme

During the game, you can *change* your shape. These upgrades make you look more and more like a *sheep*. (The round, bouncy form of sheep is the “ideal” shape for this game.)

You are all fleeing from a wolf. That’s why you’re running.

## Objective

Be the first to reach the finish line

## Control

Multiple possibilities:

* You roll automatically, press/hold a button to slow down.
* You roll automatically, press a button to reverse direction
* Use *two* buttons: one to roll forward, one to roll backward
* Use the *joystick* to roll/push in a specific direction

## Terrain types

* **Finish:** touch it to win
* **CoinLock** =>backdrop for coin lock
* **Reverse Gravity**
* **No Gravity**
* **Speed Boost**
* **(Speed slowdown? Speed reset? Slowmo?)**
* **Glue** => touching an old part of yours will glue it back to you
* **Spiderman** => you cling (strongly) to all walls around you

## Room types

**Make these modules.** Instead of putting them *inside the room script*, just spawn an extra node with its own script. Once fulfilled, it just sends a signal (to its parent room) that it should release its lock).

* **CoinLock:** Regularly spawns new coins. (Within min/max bounds.) When touched by player, its collected. Keeps counter in background. When counter above 10, the lock opens.

# To Do

## Gameplay (Essentials)

**Step 1:** When somebody finishes,

* *Save their rank.*
* *Disable their body*. (They are still visible, though transparent, and can move. But won’t interact with anything.)

**Step 2:** Once *everyone* has finished, show an overview of the ranks + times on screen.

* Don’t go to a separate screen; just show in a neat widget to the side.
* Allow restarting. (Or going back.)

## Movement (Essentials)

**Step 1:** Reduce RayCast length to whatever is needed. (When updating a body, always calculate its *outer bounds* (max coordinates from centroid. Make raycast as long as that, and then *slightly longer*.

## Pathing improvements

**Step 1: Prefer going in the same direction as you are**.

* For example, if we’re building rooms to the right, prefer adding another one to the right.
* Make going up/down much less likely
* And completely reversing our direction very rare.

**Step 2:** If we’re stuck, add a teleporter. Once *all players* touch the teleporter, we simply

* Add a new room *completely somewhere else* (also with a teleporter terrain)
* Move everyone there
* Allow the algorithm to catch up and path from there again.

More detailed:

* When teleporter is placed, toggle “pause\_room\_generation” to true (which forbids new rooms.)
* Paint the terrain to be a teleporter.
* Whenever a player enters teleport terrain, *recheck* all players. If they’re all there …
* Place a new room *completely somewhere else*. Unpause room generation. Teleport all players to the center of the new room.

## Map creation

**Step 1:** After a certain distance, add a “lock”.

* Basic Lock: Spawn multiple coins or buttons. After you’ve collected them *all*, the lock opens.

The idea for locks:

* Place solid bodies on the *edges* of the whole room, except the side we came from.
* Paint the whole room a certain terrain.
* Now *use* the content of the room for the “minigame” for the lock.
* (Once the minigame is completed, all those *edges* around the room disappear. And perhaps the locking background fades away.)

## Player creation

**Step 1:** Dynamically add players.

* Give them their corresponding inputs.
* Place them at the start of the level
* Give their own **color** (used for all parts of them) and make sure sliced parts **copy the player num of their parent**.

**Step 2:** Start players with **predefined shapes**. (Probably looks cleaner and is more functional.)

* Circle
* Square
* Triangle
* “L”-shape
* Pentagon / Hexagon
* Parallelogram
* Star shape
* “klavertjevier”

## Movement Improvements

**Step 1:** Limit 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*.

## 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.

# 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.