# Game Off (2021)

Theme: **BUG**

Deadline: **1 December**

Requirements:

* Mostly new code/art/ideas for jam
* Put whole project on GitHub (open source)

Soft requirements:

* Single player mode available => most people won’t be able to test with multiple
* Web build available => just way easier and platform-independent

## Idea

**Everyone is a *spider* stuck on a (cob)web.**

This means you can only move over the current lines in the web. Obviously, you can *change* this web by shooting new lines or breaking old ones.

Is it **competitive** or **cooperative?**

* If competitive, it might be hard to create a solo player variant.
* If cooperative, it might be hard to find a good common goal.
* It’s also possible to make it a puzzle game … but do I really want that? (Could also try both … )

## Movement

You can point in all directions. The system will find the line closest to your chosen angle and move along that.

If you press jump, you jump in the direction you aim, landing on the first line within reach.

* If you have resources ( = silk), this creates a *new* line. (From your starting position + the point where you landed.)
* If there is no other side, or it’s too far away, you will fall off the web and die?

Resources spawn all over the place. (Sometimes on the web, but usually, on purpose, some distance away.)

* Silk => needed for players
* Bugs => points for objective
* Powerups => in one form or another (keep it themed)
* Predators =>
  + Other spiders (Black Widow)
  + <https://romneypestcontrol.com/what-eats-spiders/>

The map also contains many “fixed points”. These can be used to attach lines, but are also a special location (such as a safe resting place, the default spawn for a predator, etcetera)

**IDEA:** A *water* map, where the *points* that connect the lines drift and float with the water. (And landing somewhere sends a shockwave around you.)

**IDEA:** A wind/dust map that can blow you off the web at certain points, causing you to react quickly or die?

**IDEA:** Certain types of silk that are **locked to a certain player.** ( = forbidden to others)

**IDEA:** Similarly, silk that makes you move *faster* or *slower*, silk that *wobbles based on your weight*, etc.

In short: **SILK TYPES** (are the new terrain types)

**IDEA:** To “catch (certain bugs” you simply need to trap them in the spiderweb. They will fly freely, but if they encounter a line of the web, they turn around. They have limited stamina, so if you’re able to trap them for X time, they become tired and will run into your web.

**IDEA:** Other bugs simply appear on the web immediately. (The “easy ones”.)

**IDEA:** Other bugs can be caught by jumping at them.

# Collision Layers

These are the collision layers used by physics in the game

1. Spider web (points and edges)
2. Players
3. Collectibles

# To do

**Spawning:**

* Don’t alternate blindly => count how many items are ON the web and how many are OFF (they know that when placed).
* Whichever is lower, place one of those.

**Effects:**

* When jumping, make the new line appear *gradually* (out of our butt :p)
* Item pop-up and removal tween
* Silk change tween
* Feedback

## Optional

**Spider animation (Improved!):**

* <https://www.youtube.com/watch?v=e6Gjhr1IP6w>
  + Yes, interpolated movement
  + *Start* the legs in a zigzag movement
  + Only move a leg if the others (“supporting ones”) are grounded.
* <https://www.youtube.com/watch?v=LNidsMesxSE>
  + GDC talk about it, might be interesting in any case
* Do a *intersection check* to find any surfaces near that area, then reset to a point on them?
* *Interpolate* the resetting (instead of making it instant)?
* How to ensure legs go in alternating patterns?
  + Maybe *queue* resets. Each frame, check the queued resets. We only allow it to continue, if the surrounding legs are in the right position.
  + Example: a leg wants to reset. Then the legs before and after it should be reasonably far forward (low distance). The leg on the other side should be reasonably far forward as well.

**Manual starting levels (instead of randomized)**

* Or build all levels from the “shoot line” principle.
  + In the editor, create Line2D nodes.
  + Upon load, convert their end points to actual points. (Snap to existing ones, so I can be imprecise.)
  + And then connect the edges.