Theme: Unconventional Weapon

Design:

- The weapon is being AI: you hack pieces on the map while not controlling most of the units that need to win. Alter map passability, etc
- ZK Dominatrix game. Every unit is a dominatrix. Capture trees!
- Missiles game. Every unit is a tacnuke. Make econ by launching econ missiles, etc.
- Forcefield weapon
- **Gravity weapon game:**
    - Attract target
    - Repulse target
    - Extra:
        - AoE attract/repulse
        - Black hole/reverse black hole (attract/repulse things, including projectiles, to an area that you mark)
        - Make a single target or aoe float above ground
        - (The previous two might require you to throw the weapon, which would make you disarmed for the time being)
    - Sci fi-setting.
    - Use the environment to kill enemies (push them to hazardous areas)
    - Or just impart sufficient impulse to smash them against ground or each other
    - Take advantage

Task priority

Tasks are separated in milestones, and milestones should be done in a predefined sequence.
Assign yourself to the task you're currently doing. Once you've completed the task, you can also use Mumble to inform other members. Tasks (Github issues), can be closed either manually or by writing "close[d]/fixe[d/s] #number" in commit messages (either way is fine).
You can make new tasks/milestones for either yourself of other people. The idea is to get the game running ASAP, and with that in mind we should aim to have it playable after day one, and use the rest to add textures, animations, effects, missions, improve gameplay, polish everything and solve various other minor issues.

The current general TODO list is:

- Preparation (before the jam): https://github.com/SpringCabal/GameRepo/issues?q=is%3Aopen+is%3Aissue+milestone%3APrep
- Placeholders (immediately once we figure out the idea): https://github.com/SpringCabal/GameRepo/issues?q=is%3Aopen+is%3Aissue+milestone%3APlaceholders
-

Availability timetable

Please fill your best estimate. Times are in UTC, 24h clock
If you are unsure about your availability, state that as well (it's better to underestimate than to overestimate). We will try to plan our workload based on this.
If your status changes during the jam, update this so everyone's aware.

	Anarchid	gajop	hokomoko	Picasso	Silentwings (approx)
Saturday 01:00-	01:00 - 02:00 05:00 - 21:00 (with breaks)	01:00-16:00	04:00-19:00	6-8 12-16	9.00-21.00
Sunday	07:00 - 21:00	01:00-16:00	17:00-21:00	4-6/ 12-15	9.00-21.00
Monday	05:00 - 7:00 16:00 -	01:00-14:00 22:00-	05:00-17:00 ? 22:00-	8-9 16-10	-
Tuesday -02:00 (extra hour is for packaging)	- 02:00	-02:00	-02:00		-

Checkpoints

Checkpoints are scheduled roughly once every 3 hours and will be done over voice comms. Mumble (text and voice) is used for communication during the entire competition, so people are expected to be in the channel during their active time slots.
Remember, communication is absolutely key to achieve good results.

Stuffs we're gonna have

Enemies (robots?):

- short-range beam cutter (gravity has no impact): https://github.com/SpringCabal/Gravitas/issues/13
- projectile gun: https://github.com/SpringCabal/Gravitas/issues/21
- slow, tracking missile (use gravity to stop it): https://github.com/SpringCabal/Gravitas/issues/26
- attract/repulse gun: (similar to https://github.com/SpringCabal/Gravitas/issues/12)
- Bombs (like ZK roaches) that you throw on other units?

Player weapons

- attract/repulse gun https://github.com/SpringCabal/Gravitas/issues/12
- player jumping https://github.com/SpringCabal/Gravitas/issues/28

Environment

- walls (destructible/indestructible): https://github.com/SpringCabal/Gravitas/issues/29
- simple crate for moving stuff: https://github.com/SpringCabal/Gravitas/issues/30
- explosive barrels (explode when shot or put in fire) https://github.com/SpringCabal/Gravitas/issues/15
- objects on fire (sets other things on fire and kills units) https://github.com/SpringCabal/Gravitas/issues/14
- electricity field (electrocutes things, has pulses) (it should stun robots and do damage to the player): https://github.com/SpringCabal/Gravitas/issues/31

Maybe:

- menacing spikes (some units die when close)
- mines (explode when units come close)
- void ground.. with air/unit suction
- turrent enemy units (can't be moved)
- player uplift ability (pull all units in air)
- player sphere (black/white hole) gun

Tasks: Art

Name	Comments	Status
Placeholder models- - player character (marine) - enemy - mine - barrel	We can just use the Corruption ones (already added most). New placeholders make sense only if different scale (e.g. barrel)	done
Terrain textures		kinda done
Actual models -	Maybe the player character could have gravity guns instead of hands?	robot: missing character: needs texture
Animations		character: basic ones done robot: none
Unit Textures		done for statics missing for chars

Progression(story)

- Pull a crate
- Push enemies into fire
- Use exploding barrel
- (Misc)
- Jump
- (Misc)
- Pull enemies down
- (Misc)
- (Stun)
- (Misc)
- (Special/End)

Post Mortem

Team (summarized):

- The Good:
  - Working as a **team** was fabulous. Making this game would've been impossible without utilising the knowledge, expertise and abilities of each member.
  - The **warmup** was extremely crucial for understanding the work process, the scope of what we can achieve in 2-3 days and getting to know the team members and the resulting dynamics.
  - **Github issues** were extremely useful in keeping track of what works and what doesn't.
- The Bad
  - We had very little time to playtest.
  - Too much time was wasted dealing with the engine's (limited) collision system, and it caused the majority of the issues players experienced with the final game.

ashdnazg:

- The Good:
  - Working as a **team** was fabulous. Making this game would've been impossible without utilising the knowledge, expertise and abilities of each member.
  - The **warmup** was extremely crucial for understanding the work process, the scope of what we can achieve in 2-3 days and getting to know the team members and the resulting dynamics.
  - **Github issues** were extremely useful in keeping track of what works and what doesn't. More labels (art/code/etc.) could could probably make it even better
- The Bad
  - I think picking a **genre** (Action/Puzzle) that doesn't fit the **engine** (RTS) caused 90% of our issues. While it does make the entire development more interesting (which is very important!) it doesn't make the gameplay better (which is a tad bit more important). When using a specialised engine, picking a closer genre may result in better games.
  - **Music** could use some improvement probably. I didn't have any muse so I just made something simple and put it in low volume.

gajop:

- The Good:
  - Warmup was essential, we wouldn't be able to do half of what we did if it weren't for it.
  - Github issues were great and so was mumble.

    - Most of the time people handled their responsibility well. There were very few cases when person's A work conflicted with what person B was working, and you could rely on other people's stuff to work.
    - The end result showed many new possibilities in Spring and was drastically different than any of the currently available games.
    - We made a game that I actually enjoyed playing even though I made the missions.
- <u>The Bad</u>
    - There were too many issues with the Spring collision system. Getting the walls to be blocking while not completely ruining pathing and impulse and having them displayed seemed impossible. This wasted much time and is still unresolved and causes the game to be jittery (it also had a huge impact on balance and made it much harder than it was originally designed to be - when there were no wall collisions implemented).
    - The full game and level design should have been written in advance or at least updated.
    - There was some miscommunication in the art requests. We ended up having duplicate gate textures, unused ground decals + textures, unused barrels and even a boss fight that wasn't well designed. All this was done while the first robots you encounter in the first 5 minutes of playthrough didn't have death animations (the same goes for exploding walls).
    - A better tested engine (this one seems to have ground rendering issues) as well as support for multiple specs: Low/High and an optional Medium spec as well. Different .sh and .bat files would suffice, but this needs some prior testing.
- <u>The Bad (minor)</u>
    - Maybe a single top-level repository could have been used instead of this many. Most team members didn't get to test the mission until late in day 2, while some didn't do it until release. This should've been simplified yet still allowed for some degree of separation.
    - Slight improvement in the tools (Scened: generate build icons + better handling of /luaui + /luarules reload), as well as a better pre-existing game setup. Scened could also be made to generate .sdds as project files instead of custom project dirs, and having it enable/disable widgets (through the widget handler) instead of writing custom "if devMode then .." would allow for better modularization
    - Engine needs to be extended to allow for in-game editing of specular, normal as well as diffuse void maps.
    - We still had damage happening by pushing stuff. This was against the mission design and completely screws up balance (there are robots that are immune to fire and are supposed to be either thrown into stun fields or killed by throwing them down from cliffs, which can be circumvented by just pushing them to each other, and that trivializes it).

- Should have waited with polishing the map until after the final models have been made. Many of my changes turned out to be unnecessary due to the different models (width and shape).

silentwings:

- The Good:
  - We worked together very well.
  - Our skills were a good spread across what was needed.
  - Github+mumble was an effective mix of ways to communicate.
  - The warmup weekend gave us some handy code & an idea of what was realistically achievable.
- The Bad
  - It was a shame to lose quite a bit of time to an engine bug.
  - We had very little time to playtest, although idk how we could have found time.

Anarchid

- The Good
  - We have met our objectives and answered all the challenges, proving our divine mastery of the Spring RTS engine.
  - We have shattered the dominant paradigm in 72 hours. The game doesn't even **look** like you expect a Spring game to look.
  - I gained the ability to use Mumble. Awesome!
  - Warmup was essential. Dragging our specialized tools to answer challenges was too.
  - Teamwork occured and was fun.
  - I learned to make textures from scratch. That stuff isn't even *that* hard.
  - Being actually able to use non-manually-coded-but-artsily-created-in-blender animations in a live project all working flawlessly from the first attempt felt like a Crowning Moment of Awesome.
- The Bad
  - Spring physics suck. The same game done in Godot would have been faster and better.
  - There was a ton of time lost on figuring out various bugs and the voodoo that is Spring collision volumes
  - The controls are really unintuitive and we could have spared time to fix that, completely abandoning the RTS tropes.
  - Despite having an intel gpu person in the group, we didn't have any fallbacks for fire and electricity
  - Way too few particles effects in general.
- The Ugly
  - 5 robots in the first room is enough to kill people, and yet gajop complains that they take damage from being splattered on walls!

- We had two different art pipelines and two different animation code systems, making fixing stuff sometimes impossible.
- That Crowning Moment Of Awesome? Is just par for the course everywhere else outside of Spring.
- My computer melting down meant i had to use one with an Intel GPU -> we had no shader textures on most of the models. Speculars and emission would have added a *lot*.

Picasso:

-The Good:

-Realized on first day afternoon that it actually is LD (- as in at this very moment, where the fuck are you!)

-Springs Metall looks still awesome, and shows that good default shaders can save a day.

- ToolKit needs a splitup by purpose
    - Functions in Toolbox proofed reliably reusable which is nice..
    - Don t remember much of the texturing was really groggy then
    - Made some nice Giger Trees

-The Bad

-Didn’t quite grasp the whole warmup concept. Actually though that we would continue with corruption.

-Everyone in Spring has his own approach. Approaches dont mix really well.

-Physics. Should have put a bullet into spring ages ago. also advanced Physics not for all units.

-Botched the unitscript- Unitscripts should have a threadnumber limit.

-Still cant blender well

Future

We should make some plans for the story and general mechanics of the game. Things will certainly need some rework (especially the mission) to fit both the story and mechanics.

Story

Who/what's Gravit? Human/Living being/Robot/Cyborg? What's his/her story? Are there multiple Gravit out there? Who/what are the enemies (robots)? Where are we located? What's the year/time/setting? Should it really be called "Gravit"? What about the enemy naming?

Gameplay

Ideally the story and scenario design should allow for multiple Gravits (although perhaps not all looking the same - see L4D). Some scenarios could be designed to be played only in multiplayer, either in co-op or PvP. Multiplayer (co-op) should be designed well so it's fun. Maybe Gravits could resurrect each other by shooting with their beam on fallen members? Scenarios could be designed in such a way so that they can automatically scale for a larger number of players.

Gravit should have a small number of abilities and all of those should be usable with the gravity gun and have some impact on it. Jump is an example of such a mechanic, that allows Gravit to pull objects in the air while doing it. Something that might give a temporary speed boost may also be interesting. I think we should aim at around 4 abilities: Q for gun switch, W for jump, E for something new and R for abilities specific to the scenario.

Scenarios

A wide variety of scenarios can be designed, but what we should consider here is the basic mechanics that we wish to utilize in all of them.

- Damaging units by collision. The damage done should depend on what both objects are: based on the force of the impact, but also their density (e.g. small, high mass objects like bullets will usually deal a lot more damage). Fall damage should also be calculated using this mechanic. There should be a difference between elastic and nonelastic types of impact.
- Pushing units into/out of areas. This should be the preferable way of dealing with units, and for some encounters it might be necessary to use this. Some areas might also provide bonuses for the unit (e.g. by regenerating its health or preventing it from taking damage). Pushing units into the void is a special case of this.
- Detonating explosives. Using explosives can be pretty hard and risky, so it should also be rewarding based on this. Explosives are also one of the few simple AoE ways of damaging units.

- Plates mechanic as a way of impacting stuff. Plates are fairly easy to understand and can be used in a wide variety of ways to control encounters, not just limited to gates. An example of this would be using plates to open/close fire fields.
- Crates and other misc. objects. Crates can be used to impact plates, but could also be used as a cover or a way to damage units. It might even be good to stick to just crates and maybe some other simple geometric primitives such as balls that could be used differently, and just provide different textures/characteristics (e.g. size or mass).
- Moving platforms or elevators. This might be a limitation of Spring currently, but it could be interesting if some platforms (on which units can walk on) were to move. It could be doable with movectrl though, with some amount of accuracy.
- Enemies. Properties that could be interesting:
    - Bullet dodging
    - Rocket deflecting
    - Close-range avoidance
    - Immobile enemies
    - Enemies with gravity beams (either Pull/Push or both)
    - Enemies unaffected or healed by certain attacks (e.g. immune to fire or collision doing no damage)
    - Enemies that have phases (e.g. in phase 1 they're immune to fire, but in phase 2 they're immune to the electric field)
    - Slow-rotating turrets
    - Slow reload enemies
    - Enemies that work in pairs (or multiplies), and that are strong (or weak) when they're close to each other, or when the link between them is not blocked

Tutorials should be designed in such a way that players are forced to use the skill that we're trying to teach them there. For example, if the player has already used "Push", and we're teaching him how to use "Pull", then the tutorial should be made so that it's impossible to solve it via "Push". One way of doing that is to separate the Crate + Plate from the player via the void, to disable jump, and to place the two objects in such a way that they can't be pulled from the other side.

Tutorials don't need to feel like they're tutorials, nor do they need to follow in sequence. It's also possible to leave some advanced tutorials (like Pulling + Pushing units in air) for later parts of the game.

Enemies and encounters should always be designed to look in such a way that it's clear what they do. Example: It's reasonable to presume that an enemy that's emitting fire is immune to it.

Encounters should if possible be designed by Learning via Example. For example: A robot that walks towards you through fire and gets destroyed, gives you a hint ..

Gravity beams shoving robots into Fires can be hints.

Portal used this to great effect. You always see things before you can use them in action.