=-=-=-

1. Cubes ... racing ... in space ?

Space cube racers is a racing game located in space. Space can be a problem since there is no ground, no air, no gravity as well as incredibly cold and empty and overall not very exciting. Except that now, in this year of 21XX, that emptiness has been decreased by adding racing tracks and racing ships to race in those tracks. Which makes space much more exciting.

Your ship will provide you air, mildly warm air. That means you can actually be in space, in your ship and not die of cold, or lack of pressure. Which is good.

But, I hear you say, outside of space, there is not a lot of air, well none to be pedantic, and controlling a ship in space must be extremely hard and frustrating, right ?

Wrong, because of this new technology: cube ships. A cube ship have 6 thrusters, one on each face. Inside that cube is a spherical cockpit from which you can pilot the ship. That sphere will turn inside the cube, which in turn will rotate the outer cubic frame in the opposite direction (because of some funny law of physics stating that for each action there is an equal and opposite reaction. And rotation is such an action).

All that is monitored by a smart computer that knows which thrusters to fire to move your ship in the direction you are pointing. And this smart computer will then ensure that every time you turn, the ship moves toward the direction you are facing.

That makes navigating in space a child's play. Well maybe not a child's play... but at least a lot less frustrating and hard. You will still have to show some skill... after all this is what racing is all about.

Oh. I forgot one minor point: in space you can go as fast as you want. **But don't**. How retarded is that to have this immense power to not use it? Well, as the wisdom says: "speed without control is nothing". And the faster you go, the more difficult will it be for your ship to turn. Which is a problem.

After this small introduction you are equipped with sufficient knowledge to show the universe that you have the skills to Space Cube Race.

2. Some warnings

This game have a lot of flashing effect due do getting through many gates very quickly. This helps provide a good sensation for speed, but will be problematic to people prone to epileptic seizure from visual stimuli. **EPILEPTIC PLAYERS BEWARE**. Better be safe than sorry.

This game has been designed to be primarily used by gamepads with good analog sticks. As it is a PC game, you can use your keyboard and mouse, but really, this is less than optimal. Also, joysticks can be used, but by personal experience, beware of tendonitis. Even though I play some space sim with a joystick without any problem, this game is too fast paced and even a short session tends to hurt my wrists.

This game has been developped and is being maintained by a single person. This means you should expect some problem with the game, especially with your controller. While I tried to use a very inclusive method to handle controller, I do not have many type of controller to test and have them work out of the box. I provided you with all the tools to be able to configure any controller you can throw at this game, and the game *will* make its best effort to provide you with a good configuration. But expect to have to configure your controller before being able to play.

4. Minimum requirements

The game should run everywhere Java 11.0.4 runs and where the graphics card is not made of potato chips. Even an intel integrated card should be enough. The major factor being the graphics memory which should be around 200M at the minimum, as this is the amount of memory used by the tracks.

5. Controllers

Because of the input library used by the game engine, some configuration may be required for your controller to work.

The game uses an internal "Virtual gamepad" (based on a PS3 controller) that you will have to configure if your controller is not well recognized out of the box.

Here is a drawing of how things should be mapped:

```
Your controller:   (1)  Game Virtual Gamepad      (2)    In-game binding

Button X         ----->    Button Triangle       ----->    Respawn
Button Y         ----->     Button Square        ----->    Brake
Stick Left <-    ----->      Stick Left <-       ----->    Yaw Left
etc ...
```

The step (1) is done by an external tool called JoystickConfig.

The step (2) is the normal binding procedure you will find in game.

5.1 JoystickConfig

First of all, since JoystickConfig uses the same input library as the game, be sure to plug your controller before launching the tool.

5.1.1 Joystick Type

-=-=-=-=-=-=-=-=-=-

You will see this tab selected when the tool is launched. It shows a 'joystick config' list on the left and a controller list on the right.

- Select the appropriate controller
- Create a new 'joystick config'
- Rename it. You should use the scheme [Linux/Distribution] on linux, or simply [Windows] on windows

This will actually be used in-game to find an automatic match for your controller.

A note on the other buttons:

- remove: removes the selected configuration
- print: prints the selected configuration in a pop-up and in stdout (the output of the console if you run inside a console). This can be used to send me your configuration or post it in steam discussion so that I can incorporate it in the game at a later release.

- 'bind this joystick ID to the selected CFG': load the selected configuration and associate it to the selected controller. This helps test or modify the selected configuration.

5.1.2 Axis Binding

-=-=-=-=-=-=-=-=-=

This tab is made of 3 parts:

- Axis: the list of axis your controller have.

- Axis Type: an axis have a range of values it can take. This will define what that range and where in that range the axis is considered "released" (meaning the state where no force is used by the player).

- Binding: this is used to define what part of the virtual gamepad, as it is understood by the game, is to be bound to the selected axis.

First, you should move your gamepad axis (analog stick, trigger, etc.) and see which one is actually considered an axis.

Once you picked the axis you want to bind, see what the range is by going from one maximum to the other. On the part you will see something like this:

0 => none (-1.00)

when pressing fully on the left on the left stick (this is an example, it may not be that particular value), and:

0 => none (1.00)

when pressing fully on the right on the left stick. And finally:

0 => none (0.00)

when not touching the left stick at all. This means the axis 0 has a [-1, 1] range with a release at 0. This is a common range that can be found in the range list. But if you find that your axis is not a common one, use custom and fill min, max and range accordingly.

If you find that, when releasing the axis, the value varies a little bit around some value X, for example:

0 => none (X.01), 0 => none(-X.02), etc... this means you have to define a deadzone, i.e. a zone where your axis will be considered released. If you set deadzone with 0 and your axis do not hit presicely X when released, then the axis will **not** be considered released.

The precision slider is there to adjust the amount of digits displayed in the axis part. This may be useful to get a better view of a deadzone. Otherwise this should not be very useful.

Finally, once you selected an axis, adjusted its axis type, you are ready to bind it to the virtual gamepad on the right.

Depending on your axis type, you will have one or two binding to configure.

When your range is [X, Y] and:

- the release is either X or Y, then the range is not split into two range by release.

- the release is in between X, Y, the release splits the range into two ranges.

In the first case, you just have to press the button you want associated to the axis.

In the second case, you will have to select which part of range you want to bind and then click on the button you want associated to the axis.

For example, let's say you want to bind an analog stick, the release will most likely be at the middle of the range:

1) choose 'lower than release point' then "Stick <"

2) choose 'greater than release point' then "Stick >"

5.1.2 Button Binding

-=-=-=-=-=-=-=-=-=-=-

This behaves like the axis tab, but much simpler since there is no range associated to buttons. Those are simple digital events being pressed or released. So the only think to configure here is which button is mapped to which button on the virtual gamepad on the right side. To do that, select the button on the left, like for the axis, and then click on the desired button on the right side.

Note that there is a text input field "In-Game button name" where you can change the button name to a name that corresponds better to your controller.

5.1.3 Test

-=-=-=-=-=

Like its name implies, this tab is there to ensure that the bindings have correctly configured.

5.1.4 Saving the configuration

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

Just exit by closing the window. The configuration will be saved to the selected configuration. Note that the exit procedure may be a bit slow due to that saving.

5.2 In-game controls

The first time you launch the game, the controllers will be mapped by default using the knowledge of your OS and your controller name and by examining the button name data that can be looked upon using the input library. However, while this may appear sufficient in theory, in practice the name controller may be generic (PS4 using Bluetooth are called "Controller" or something like that...), the button data may not be filled in, and you may be using an OS that is exotic but still compatible with all the supported OS.

For all those reasons, the auto-binding may fail. You then will have to set which configuration (made at the previous section 5.1 JoystickConfig) is to be associated which controller.

5.2.1. Default Bindings

-=-=-=-=-=-=-=-=-=-=-

This is the first thing you should try when your controller is not working properly out of the box. Select the controller in the ID row, then select the configuration in the CFG row (and do that for each player) then hit apply. This will setup the controller to the default binding. If the binding is not to your taste, then you still need to go to the next step.

Here is the default binding:

	Keyboard:	Controller:
ThrustForward	w	Cross
ThrustBackward	x	Circle
Brake	s	Square
ThrustLeft	a	R.Stick Left
ThrustRight	d	R.Stick Right
ThrustUp	z	R.Stick Up
ThrustDown	c	R.Stick Down
PitchUp	axisUp	L.Stick Up
PitchDown	axisDown	L.Stick Down
YawLeft	axisLeft	L.Stick Left
YawRight	axisRight	L.Stick Right
RollLeft	q	L1
RollRight	e	R1
BackOnTrack	Tab	Triangle
ZoomIn	wheelUp	L2
ZoomOut	wheelDown	R2
AutoRoll	Left Shift	Select

5.2.2. Custom Bindings

-=-=-=-=-=-=-=-=-=-=-=-=

One thing of notice is that this screen has no support with controller because it would mess with the auto-detect.

Yes, that's a bit of laziness. Deal with it :-)

The first tab, "Input Types" is just a summary of the keyboard layout and the controller that were connected when the game started. The controller and keyboard layout can be changed, but this is not the prefered place to do that.

Also, you can find an activity indicator next to the controller, to find which controller is associated to which ID.

The second and third tab are equivalent but concerns two part of the game: in-game vs menu controls. This have drop-down list where you can choose your ID/CFG and trigger. Note that you can use any amount of ID/CFG for a single player. This was made because of my experience with PS3 controller (on linux) where the buttons and sticks were recognized on a controller but the accelerometer axis of the controller where assigned to separated controller, even though there was really only single controller. This has changed with time (and kernel releases) but the necessity to adapt to such situation was written in the game. So if you have a weird

controller or a bunch of different controller, you can use them any way you want.

The main appeal to this screen will be the automatic input recognition:

1) be sure the controller id / cfg is correct
2) click on the action
3) press the button / axis on the controller
4) profit !

Also to note: you can copy the controller id / cfg by using CTRL + left click. Be sure to not release CTRL as long as you want to copy. The mouse click can be released as much as you want.

6. Sound

The sound option panel may need some explanation: The music and general sfx are the main volume control. Then to have a finer control on the sound effects, there is a volume control for each of the sounds in-game:

- ship: this is the sound the engine does, heared from the inside of the ship. I usually make it barely perceptible but you may want to increase its presence. It's up to you.

- speed: this is a representation of the ship speed generated by an on-board sound generator. This helps to know your speed by an auditive cue.

- bump: the sound you make when you are screwing up by hitting the track.

- gate: the sound generated when your ship get through a gate.

7. Things to know and Troubleshooting

The game does not auto-detect your anisotropic filtering level. Please select the maximum amount for better visuals.

If an option renders your game unusable for some unforeseen reason, simply remove the 'settings.erase.to.reset' file.

This is also true for the controls and the single player progression.

On linux, the game will use the selected sound card at the moment the game is started. If you have multiple output, be sure to select the one you want for the game and set it as the global default sound card. This may apply to other OSes.

The controller library used in this game is a bit touchy with the controllers. Do not unplug a controller while playing: this may cause the controllers to stop working. If your controller suddenly stop working, I am afraid you will have to restart the game. This should be a *very* rare occurence, when I had it happened a lot - something like once an hour - I managed to pin the problem down to a cable problem.

If a controller has not been configured, be sure to plug it (or have it paired to your Bluetooth dongle) before you run the game, or it won't appear in-game.

8. Open-source <3

==========

This game would have never happened without the work of others. Here are the open-source or otherwise free tools I used:

- Blender, for the tracks (check the plug-in and make your own track !)
- Gimp, for the main screen background and fonts.
- Ardour, ZynAddSubFx, audacity, for the music and sound
- image magick, for some stuff related to blender
- spacescape (alex peterson), for the space background
- JMonkey Engine, a great game engine for java
- Kotlin, because this game is written in kotlin !
- Intellij IDEA community, a cool IDE for kotlin
- pycharm community, a cool IDE for python (used to code the blender plug-in)
- BMFont, and the fonts “air americana” from Tiago Sá (found in AUR repository of arch linux) , “cousine” by Steve Matteson (fontsquirrel), “gidole” by Andreas Larsen.

Those are the tools directly related to the game, but being that I am using linux, there is a ton more open-source software that was being indirectly used to create this game.