

# CS211

## Milestone #3

22 June 2020

For this final milestone, you should submit the tangible game that you have developed over the semester. If you have done what was expected for Milestones 1 and 2, then you only need to complete the instructions from Week 7 and Week 12. The deadline for submission is **the 22nd of June, 23:55 CEST**. This part of the project will be graded based on the items listed in this document.

## Code Submission

The submission procedure is the same outlined for Milestone 1 and 2: we ask you to submit your Processing project via a **git repo on c4science** and then upload the URL of the repo on Moodle. Specifically, **we expect your submission to contain a sketch named TangibleGame that is the main sketch of your game**. If you work with Java IDEs other than Processing, copy and paste your code in Processing, and make sure that it runs in the Processing development environment.

When your code is ready for submission, **tag your git repository** with the tag `milestone3_submit`:

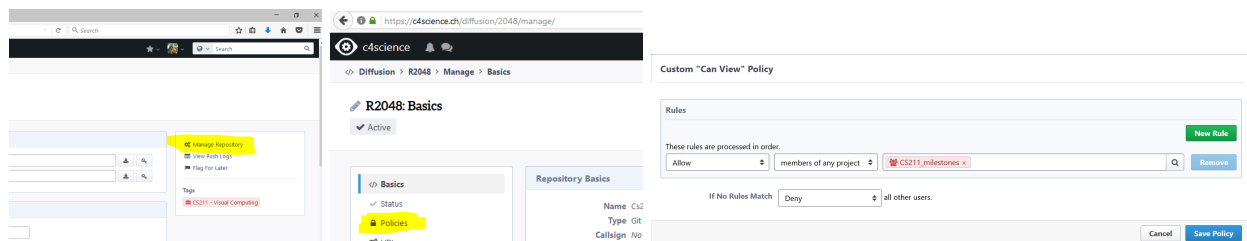
---

```
$ git tag milestone3_submit
$ git push --tags
```

---

(if you made a mistake, re-tag your repository as desired with `git tag -f milestone3_submit` and ‘force’ push it: `git push -f --tags`)

Then, submit the public URL of the repository on Moodle (under *Milestone 3*). **Make sure that we have the right to clone it!** To do so, in all your policy settings (visible, editable, pushable), define a custom policy and add `bb_bruno` (Barbara Bruno) and members of project “CS211\_milestones”.



**Note**

Even though the project is to be done in group and you will receive one single grade per team, **it is important that every team member contributes in every part of the project.** We remind you that the final written exam will also include questions related to topics that are dealt with during the project.

## Milestone 3 - Checklist

We will check your submission by running it on the video `testvideo.avi` that we made available on Moodle.

Instead of using the camera stream, **your application should take this video as input.** Tune your parameters in accordance with the lighting of the video. You can use a piece of code similar to the following to load it:

---

```
import processing.video.*;

//...

/*Capture to Movie in declaring the video class*/
//Capture cam;
Movie cam;

/*In the setup()*/
//cam = new Capture(this, cameras[63]); //If using gstreamer0.1
//cam = new Capture(this, 640, 480, cameras[0]); //If using gstreamer1.0
//cam.start();
void setup() {
    cam = new Movie(this, "testvideo.avi");
    cam.loop();
}

void draw() {
    if (cam.available() == true) {
        cam.read();
    }
    img = cam.get();
    //...
}
```


---

For more information see: <https://processing.org/reference/libraries/video/Movie.html>

**Note**

The code above uses the relative path to load the video. If you did not manage to load the video this way, try to use the absolute path. No worries, we will change it when grading your project.

When running the Processing sketch, it should:

- Show a plate at the centre of the screen, with the sphere initially at the centre of it. The plate should tilt according to the movement of the board that your code reads from the video, and the sphere should roll on it naturally.
- Show the data visualisation elements introduced in Assignment #7 (detailed below).
- Show the video augmented with detected lines and corners on a separate displaying window (the ImageProcessing window).
- By pressing , enter the *object placement mode* which allows the user to add the opponent particles emitter at the click's location on the plate (as requested for Milestone 1).



#### Note

Try to make the interaction as reliable as the methods that you learned in this course allows. However we know that because of the noise, it cannot be fully reliable.

The score-visualisation bar (as explained in Assignment #7) should:

1. show the topView 2D movement of the ball,
2. show the text box showing the current score, current velocity magnitude of the sphere, as well as the points that the user achieved in the last (hitting) event,
3. show the total score represented with the stacked-up tiny rectangles (as described in Assignment #7),
4. allow to interact with the barplot of the scores.