Kostya Leshenko
klechtch@pdx.edu

Project name: **catio**
{Cat themed platformer game and a very basic 2D physics library}

I would like to make a platformer style video game in Rust. I've never made a platformer so this will be something new for me. My daughter likes cats and has been asking me to make a video game with a cat character in it. At least the main character in this game will be a cat, this should serve as some kind of artistic direction or constraint.

At the end of the term, my project will (ideally) consist of two parts, a 2D physics library and a game with basic platformer features which uses the physics library.

Part 1 – Physics library
The physics library will implement basic game physics functionality – 2D object movement and acceleration, collision detection, and the ability for colliding objects to apply forces on each other. I might want to include some sort of tree data structure (quadtree) to accelerate collision detection. The library should be usable outside of the catio project. I am not sure what the interface is going to look like yet, but perhaps something like this might work for simple game and demo clients:

[pseudocode]

```
// Library code
pub struct World { ... }
pub enum PhysicsStatus { ... }
pub struct PhysicsObject {
    position: vec2,
    velocity: vec2,
    acceleration: vec2,
    mass: f32,
    ...
}

impl PhysicsObject {
    fn new(world: &mut World, ...) -> PyshicsObject { ... }
    fn update(world: &mut World) -> PhysicsStatus { ... }
    ...
}

// Client code
struct Cat {
    physics: PhysicsObject,
    /* other game related data members */
}
```

Note, the cat type above can instead be a more generic entity type, and physics is one of its components.

Part 2 - The platformer game
I hope to implement basic platformer game functionality. Horizontal movement and jumping for the player character (cat). Static and moving platforms. Player and platform moment, collision detection, gravity, etc. will be implemented using the physics library. Camera movement – scrolling of the game world to ensure the player character is always visible as the player moves around the world. Ability to finish the game either by finding the exit or dying from falling.

Rendering: I intend to use sprites and drawing various simple objects (platforms) programmatically.

Stretch and future goals for the game, in some order:
Double jump
Game state (main menu loop, game loops, etc.)
Sound effects
Multiple levels
Enemies
Collectable items, power ups
Sprite animation
Better visuals

SDL (https://www.libsdl.org)
Prefer to use SDL for platform independent window creation, event loop, graphics library initialization, timing, etc. I used SDL for this purpose in the past – it seems to work very well, at least in C and C++ projects. SDL2 libraries are shipped with the Steam client, and the first stable version of SDL3 was recently released. I think SDL is a good platform for indie game development (if I ever make publishable software:)

Some concerns:

Originally I was planning to use the sdl3-rs crate (https://docs.rs/crate/sdl3/0.14.5), but it seems that the SDL3_ttf module has not been implemented in this crate. I can probably implement some sort of bitmap font rendering, but it will only take more time. I think I will use the SDL2 wrapper (https://docs.rs/sdl2/0.37.0/sdl2/all.html) instead – I think since it supports all the SDL features I want to use in my project.

Bigger concern… time.