RustSweeper

Alexander Chmielus
Jasmine Cavael

Rust ABV Kurs SoSe2020
Freie Universität Berlin
Institut für Informatik

Alexander Chmielus,
Jasmine Cavael

Inhalt

* Was ist RustSweeper?
    * Erläuerung
    * Spielregeln

* Implementierung
    *Felder
    *Mienen
    *Count
    *EventHandler
    *Main

* Testspielchen

* Fragen und Anmerkungen

Alexander Chmielus,
Jasmine Cavael

Was ist Rustsweeper?

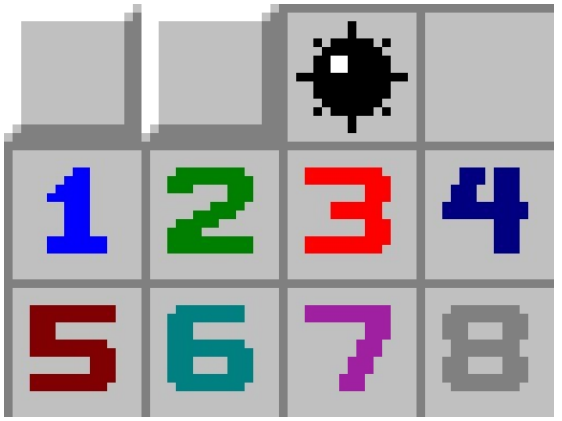
Kurze Erläuterung

* Minesweeper in Rust
* Feld anklicken
  → Nummer oder Explosion
* Nummer gibt Mienen in der Nähe an
* Gewonnen, wenn das ganze Feld frei ist OHNE, dass eine Miene explodiert ist

Was ist Rustsweeper?

Spielregeln

* Miene → Verloren
* hellblau → 1 Miene
* dunkelgrün → 2 Mienen
* hellrot → 3 Mienen
* dunkelblau → 4 Mienen
* dunkelrot → 5 Mienen
* türkis → 6 Mienen
* lila → 7 Mienen
* grau → 8 Mienen



Microsoft - Minesweeper

Alexander Chmielus,
Jasmine Cavael

Implementierung

Felder

```
pub struct Field {
    mine: bool,         //Hat Miene oder nicht
    x: f32,             //Position
    y: f32,             //Position
    counts: i32,        //Mienenzähler
    id: i32,            //Eindeutige Felder-ID
    clicked: bool,      //Angeklickt?
}
```

Alexander Chmielus,
Jasmine Cavael

Implementierung

Felder

```
impl Field {

    fn get_x(id: i32, size: i32) -> f32 {
        let x0 = 50;
        let x1 = x0 + size * (id % 5);
        return x1 as f32
    }

    fn get_y(id: i32, size: i32) -> f32 {
        let y0 = 50;
        if id % 5 == 0 {
            return (y0 + id * size) as f32;
        } else {
            return (y0 + size * (id - (id % 5))) as f32;
        }
    }
```

Implementierung

Mienen

```
        fn place_mine(&mut self) {
            let mut rng = rand::thread_rng();
            let m = rng.gen_range(0, 3);
            if m == 0 {
                self.mine = true;
            }
        }
    }
```

Implementierung

Count

```
fn get_count(state: &State, i: i32) -> i32 {

    let mut c: i32 = 0;

    if (i+1) % 5 != 0 && i+1 < 40{
    //rechtes Nachbarfeld
        let x = i as usize;
        if state.fields[x+1].mine == true {
                c = c + 1;
        }
    }
    if (i-1) % 5 != 4 && i-1 >= 0 {
    //linkes Nachbarfeld
        let x = i-1;
        let dex = x as usize;
        if state.fields[dex].mine == true {
                c = c + 1;
        }
    }
```

Implementierung

Count

```
if (i+5) < 40 {
    //Nachbarfeld direkt darunter
    let x = i as usize;
    if state.fields[x+5].mine == true {
            c = c + 1;
    }
}
if i-5 >= 0 {
    //Nachbarfeld direkt darüber
    let x = i-5;
    let dex = x as usize;
    if state.fields[dex].mine == true {
            c = c + 1;
    }
}
```

Alexander Chmielus,
Jasmine Cavael

Implementierung

Count

```
if (i+4) % 5 != 4 && i+4 < 40 {
        //Nachbarfeld links unten
    let x = i as usize;
    if state.fields[x+4].mine == true {
            c = c + 1;
    }
}
if (i-4) % 5 != 0 && i-4 >= 0 {
        //Nachbarfeld rechts oben
    let x = i-4;
    let dex = x as usize;
    if state.fields[dex].mine == true {
            c = c + 1;
    }
}
```

Implementierung

Count

```
if (i+6) % 5 != 0 && i+6 < 40 {
        //Nachbarfeld rechts unten
        let x = i as usize;
        if state.fields[x+6].mine == true {
                c = c + 1;
        }
}
if (i-6) % 5 != 4 && i-6 >= 0 {
        //Nachbarfeld links oben
        let x = i-6;
        let dex = x as usize;
        if state.fields[dex].mine == true {
                c = c + 1;
        }
}
return c;
}
```

Alexander Chmielus,
Jasmine Cavael

Implementierung

Count

```
fn set_counts(state: &mut State) {
    let mut i = 0;
    while i < 40 {
        let c = get_count(&state, i);
        let x = i as usize;
        let f = &mut state.fields[x];
        f.counts = c;
        i = i + 1;
    }
}
```

Alexander Chmielus,
Jasmine Cavael

Implementierung

EventHandler

```
impl ggez::event::EventHandler for State {

    fn update (&mut self, ctx: &mut Context) ->
GameResult<()> {
        Ok(())
    }
```

* Updated ständig das Spielfeld
* Farbwechsel der Felder muss somit hier bestimmt werden

Alexander Chmielus,
Jasmine Cavael

Implementierung

EventHandler

```
fn draw(&mut self, ctx: &mut Context) ->
GameResult<()> {
    for field in &self.fields {
        if field.mine == true {
            let rectangle = graphics::Mesh::new_rectangle(
                ctx,
                graphics::DrawMode::fill(),
                graphics::Rect::new(field.x, field.y, 50.0,
                                                    50.0),
                graphics::BLACK,
            )?;
            graphics::draw(ctx, &rectangle,
                           graphics::DrawParam::default())?;
        }
```

Alexander Chmielus,
Jasmine Cavael

Implementierung

EventHandler

```
else if field.mine == false && field.counts == 1 {
            //helleres blau bei counts=1
            let rectangle = graphics::Mesh::new_rectangle(
                ctx,
                graphics::DrawMode::fill(),
                graphics::Rect::new(field.x, field.y, 50.0,
                                                    50.0),
                graphics::Color::new(0.0, 0.0, 1.0, 1.0),
            )?;
            graphics::draw(ctx, &rectangle,
                                graphics::DrawParam::default())?;
        }
```

* hellblaues Rechteck bei einer Miene als Nachbar
* restliche Farben genau so implementiert
    → counts und color natürlich anders

Alexander Chmielus,
Jasmine Cavael

Implementierung

Main

```
pub fn main() {
    let mut fields = Vec::new();
    let mut draw_counter = 0;
        while draw_counter < 40 {
            let x = Field::get_x(draw_counter, 50);
            let y = Field::get_y(draw_counter, 10);
            let mut f = Field::new_field(draw_counter, x, y);
            f.place_mine();
            fields.push(f);
            draw_counter = draw_counter + 1;
        }
    let state = &mut State{fields: fields};
    set_counts(state);
    let (ref mut ctx, ref mut event_loop) =
ggez::ContextBuilder::new("hello_ggez", "awesome person")
        .build()
        .unwrap();
    event::run(ctx, event_loop, state).unwrap();
}
```

Alexander Chmielus,
Jasmine Cavael

Testspielchen

Alexander Chmielus,
Jasmine Cavael

Vielen Dank fürs Zuhören!

Habt ihr noch Fragen und/oder Anmerkungen?