RustSweeper

Alexander Chmielus
Jasmine Cavael

Rust ABV Kurs SoSe2020
Freie Universität Berlin
Institut für Informatik
14.07.2020

Alexander Chmielus,
Jasmine Cavael

Inhalt

* Was ist ggez?
    * Basic Template
* Entwicklung & Erfahrungen
    * Anfängliche Schwierigkeiten
    * Weiterer Verlauf
    * Typische Fehlerquellen
    * Weiterhin offen sind…
* Was ist RustSweeper?
    * Erläuterung
    * Spielregeln
* Implementierung
    *Felder
    *Mienen
    *Count
    *EventHandler
* Testspielchen
* Fragen und Anmerkungen

Alexander Chmielus,
Jasmine Cavael

Was ist ggez ?

* Rust library für einfache Spieleentwicklung
* ggez – Good Game Easily
* 2D Games
* API basierend auf der LÖVE Framework
* beinhaltet:
    * basics * sound * drawing
    * resource loading * event handling
* keine „millionenfachen “ Plugins etc. Benötigt
* Jedoch keine größeren Funktionalitäten wie Physic Engines
* eigene Libraries erlaubt

RustSweeper

Alexander Chmielus,
Jasmine Cavael

* Was ist ggez?
    * Basic Template
* E & E
    * Schwierigkeiten
    * Weiterer Verlauf
    * Fehlerquellen
    * Weiterhin offen sind...
* Was ist RustSweeper?
    * Erläuterung
    * Spielregeln
* Implementierung
    *Felder
    *Mienen
    *Count
    *EventHandler
* Testspielchen
* Fragen und Anmerkungen

Was ist ggez ?

Basic Template

```rust
use ggez::{graphics, Context, ContextBuilder, GameResult};
use ggez::event::{self, EventHandler};

fn main() {
    // Make a Context.
    let (mut ctx, mut event_loop) = ContextBuilder::new("my_game", "Cool Game Author")
                .build()
                .expect("aieee, could not create ggez context!");

    // Create an instance of your event handler.
    // Usually, you should provide it with the Context object to
    // use when setting your game up.
    let mut my_game = MyGame::new(&mut ctx);

    // Run!
    match event::run(&mut ctx, &mut event_loop, &mut my_game) {
        Ok(_) => println!("Exited cleanly."),
        Err(e) => println!("Error occured: {}", e)
    }
}

struct MyGame {
    // Your state here...
}

impl MyGame {
    pub fn new(_ctx: &mut Context) -> MyGame {
        // Load/create resources such as images here.
        MyGame {
            // ...
        }
    }
}

impl EventHandler for MyGame {
    fn update(&mut self, _ctx: &mut Context) -> GameResult<()> {
        // Update code here...
        Ok(())
    }

    fn draw(&mut self, ctx: &mut Context) -> GameResult<()> {
        graphics::clear(ctx, graphics::WHITE);
        // Draw code here...
        graphics::present(ctx)
    }
}
```

Entwicklung & Erfahrungen

Anfängliche Schwierigkeiten

* lustigerweise größtes Problem: Rechtecke malen!
  → konnte nicht verschoben werden, da Spielfeld und Spiel darauf basieren!
* hat einiges an Zeit gedauert, bis Prinzip verstanden wurde
* Folgen:
  * sehr sehr viel Korrektur
  * komplette Deadlineliste verworfen

Entwicklung & Erfahrungen

Weiterer Verlauf

* Mehrere Rechtecke → Spielfeld
* Wie wird die Anzahl der Mienen berechnet?
* Wie stellen wir die Anzahl dar?
* Wie funktioniert die Mauseingabe?
* Wie beenden wir das Spiel?

RustSweeper

Alexander Chmielus,
Jasmine Cavael

Entwicklung & Erfahrungen

Typische Fehlerquellen

* Verwechslung von Sprachen
* Vergessen von ;
* Verschrieben
* Rückgabewerte in falscher Schleife
* Wie ging das mit dem Modul nochmal?

Entwicklung & Erfahrungen

Weiterhin offen sind…

* Was ist ggez?
    * Basic Template
* E & E
    * Schwierigkeiten
    * Weiterer Verlauf
    * Fehlerquellen
    * Weiterhin offen sind…
* Was ist RustSweeper?
    * Erläuterung
    * Spielregeln
* Implementierung
    * Felder
    * Mienen
    * Count
    * EventHandler
* Testspielchen
* Fragen und Anmerkungen

* Felder durch Umrandungen voneinander abtrennen
* Feld ohne Mienennachbar bleibt auch nach Klicken weiß

Was ist Rustsweeper?

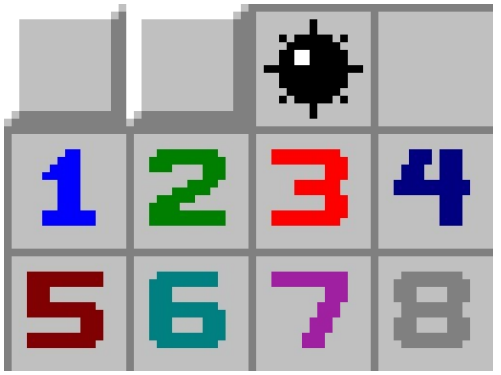
Kurze Erläuterung

* 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

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?
}
```

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, 5);
        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;
    }
}
```

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;
}
```

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;
    }
}
```

Implementierung

EventHandler

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

    fn update (&mut self, ctx: &mut Context) ->
                                                GameResult<()> {
        if !self.game_over && !self.game_won{
            if self.check_field() {
                self.game_won = true;
            }
        }
        if self.game_won {
            self.game_over = true;
        }
        Ok(())
    }
```

Implementierung

EventHandler

```
fn mouse_button_down_event(&mut self, ctx: &mut
                           Context, button:
                           ggez::input::mouse::MouseButton,
                           x: f32, y: f32) {
    for field in &mut self.fields {
        if (field.x + 50.0) > x && x >= field.x &&
           (field.y + 50.0) > y && y >= field.y {
            field.clicked = true;
            if field.mine == true {
                self.game_over = true;
            }
        }
    }
```

Implementierung

EventHandler

```
        if self.game_over && !self.game_won {
            println!("Miene Getroffen, Game Over.");
        }
        if self.game_won{
            println!("Gewonnen!");
        }
    }
```

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())?;
        }
```

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

Implementierung

EventHandler

```
        else {
            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())?;
                }
            }
        }
    }
            graphics::present(ctx);
            Ok(())
```

Alexander Chmielus,
Jasmine Cavael

Testspielchen

RustSweeper

Alexander Chmielus,
Jasmine Cavael

* Was ist ggez?
    * Basic Template
* E & E
    * Schwierigkeiten
    * Weiterer Verlauf
    * Fehlerquellen
    * Weiterhin offen sind…
* Was ist RustSweeper?
    * Erläuterung
    * Spielregeln
* Implementierung
    *Felder
    *Mienen
    *Count
    *EventHandler
* Testspielchen
* **Fragen und Anmerkungen**

Vielen Dank fürs Zuhören!

Habt ihr noch Fragen und/oder Anmerkungen?

Quellen und Hinweise

* https://github.com/ggez/ggez
    06.07.2020, 13:31 Uhr
* https://github.com/ggez/ggez/tree/master/examples
    06.07.2020, 13:32 Uhr
* https://docs.rs/ggez/0.5.1/ggez/#modules
    06.07.2020, 13:34 Uhr

Unsere Arbeit findet ihr hier:
* https://github.com/Sanuye/RustSweeper
    06.07.2020, 13:36 Uhr