

RustSweeper

Alexander Chmielus
Jasmine Cavael

Rust ABV Kurs SoSe2020
Freie Universität Berlin
Institut für Informatik



Inhalt

- * Was ist ggez?
 - * Features
 - * Betriebssysteme
 - * Benutzung
 - * Basic Template
- * Was ist RustSweeper?
 - * Erläuterung
 - * Spielregeln
- * Implementierung
 - *Felder
 - *Mienen
 - *Count
 - *EventHandler
 - *Main
- * Testspielchen
- * Fragen und
Anmerkungen

Was ist ?

* Was ist ggez?

- * Features
- * Betriebssysteme
- * Benutzung
- * Basic Template

* Was ist RustSweeper?

- * Erläuterung
- * Spielregeln

* Implementierung

- *Felder
- *Mienen
- *Count
- *EventHandler
- *Main

* Testspielchen

* Fragen und Anmerkungen

- * 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

Was ist ?

Features

- * Dateisystemabstraktion → Ressourcen aus Zip-Dateien und/oder Ordnern laden
- * 2D-Rendering basierend auf gfx-rs
- * Laden und Abspielen von .ogg-, .wav- und .flac-Dateien über die Rodio-Kiste
- * TTF-Schriftwiedergabe mit rusttype und glyph_brush
- * Schnittstelle zur Behandlung von Tastatur- und Mausereignissen
- * Config-Dateien für Engines und Game Setting
- * Timing- und FPS-Messfunktionen
- * Mathematikbibliothek mint
- * erweiterte Grafikoptionen:
 - * Shader
 - * Sprite-Batches
 - * Rendering-Ziele

- * Was ist ggez?
 - * Features
 - * Betriebssysteme
 - * Benutzung
 - * Basic Template
- * Was ist RustSweeper?
 - * Erläuterung
 - * Spielregeln
- * Implementierung
 - *Felder
 - *Mienen
 - *Count
 - *EventHandler
 - *Main
- * Testspielchen
- * Fragen und Anmerkungen

Was ist ?

Betriebssysteme

- * Was ist ggez?
 - * Features
 - * **Betriebssysteme**
 - * Benutzung
 - * Basic Template
- * Was ist RustSweeper?
 - * Erläuterung
 - * Spielregeln
- * Implementierung
 - *Felder
 - *Mienen
 - *Account
 - *EventHandler
 - *Main
- * Testspielchen
- * Fragen und Anmerkungen

- * komplett unterstützt:
 - * Windows * Linux
- * inoffiziell unterstützt, klappt dennoch:
 - * Mac * iOS
- * in Arbeit:
 - * WebAssembly
- * bisher nicht offiziell unterstützt:
 - * Android
- * für mehr Informationen:
<https://github.com/ggez/ggez/blob/master/docs/BuildingForEveryPlatform.md>

Was ist ?

Benutzung

- * benötigt rustc >= 1.36
- * auf crates.io verfügbar
- * besteht aus drei Hauptkomponenten:
 - * Kontextobjekt → enthält den gesamten Status für Schnittstellen der Hardware
 - * EventHandler → implementiert Nutzer, um Rückrufe für Ereignisse zu registrieren
 - * Untermodule → damit etwas erledigt werden kann → Funktionalität
- * allgemeines Muster: Struktur erstellen, die Spieldaten und EventHandler vereinen
 - Kontextobjekt aus ContextBuilder aufrufen
 - event::run() mit Context und einer EventHandler-Instanz aufrufen
 - Hauptschleife des Spiels somit aufgerufen

- * **Was ist ggez?**
 - * Features
 - * Betriebssysteme
 - * **Benutzung**
 - * Basic Template
- * Was ist RustSweeper?
 - * Erläuterung
 - * Spielregeln
- * Implementierung
 - * Felder
 - * Mienen
 - * Count
 - * EventHandler
 - * Main
- * Testspielchen
- * Fragen und Anmerkungen

Was ist ?

Basic Template

```
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("aieeee, 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 occurred: {}", 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)
    }
}
```

- * Was ist ggez?
 - * Features
 - * Betriebssysteme
 - * Benutzung
 - * **Basic Template**
- * Was ist RustSweeper?
 - * Erläuterung
 - * Spielregeln
- * Implementierung
 - *Felder
 - *Mienen
 - *Count
 - *EventHandler
 - *Main
- * Testspielchen
- * Fragen und Anmerkungen

Was ist Rustsweeper?

Kurze Erläuterung

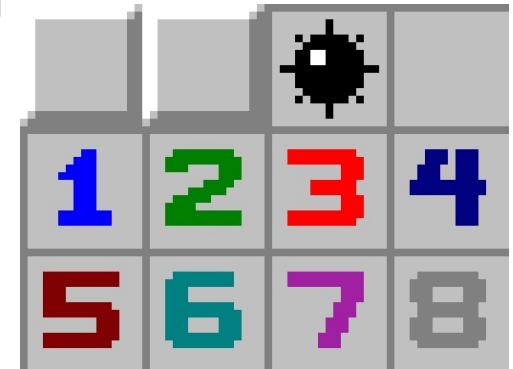
- * Minesweeper in Rust
- * Feld anklicken
 - Nummer oder Explosion
- * Nummer gibt Mienen in der Nähe an
- * Gewonnen, wenn das ganze Feld frei ist OHNE, dass eine Miene explodiert ist

- * Was ist ggez?
 - * Features
 - * Betriebssysteme
 - * Benutzung
 - * Basic Template
- * Was ist RustSweeper?
 - * Erläuterung
 - * Spielregeln
- * Implementierung
 - *Felder
 - *Mienen
 - *Count
 - *EventHandler
 - *Main
- * Testspielchen
- * Fragen und Anmerkungen

Was ist Rustsweeper?

Spielregeln

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



Microsoft - Minesweeper

- * Was ist ggez?
 - * Features
 - * Betriebssysteme
 - * Benutzung
 - * Basic Template
- * Was ist RustSweeper?
 - * Erläuterung
 - * **Spielregeln**
- * Implementierung
 - *Felder
 - *Mienen
 - *Count
 - *EventHandler
 - *Main
- * Testspielchen
- * Fragen und Anmerkungen

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?  
}
```

- * Was ist ggez?
 - * Features
 - * Betriebssysteme
 - * Benutzung
 - * Basic Template
- * Was ist RustSweeper?
 - * Erläuterung
 - * Spielregeln
- * **Implementierung**
 - * **Felder**
 - * Mienen
 - * Count
 - * EventHandler
 - * Main
- * Testspielchen
- * Fragen und Anmerkungen

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;  
        }  
    }  
}
```

- * Was ist ggez?
 - * Features
 - * Betriebssysteme
 - * Benutzung
 - * Basic Template
- * Was ist RustSweeper?
 - * Erläuterung
 - * Spielregeln
- * **Implementierung**
 - * **Felder**
 - * Mienen
 - * Count
 - * EventHandlerer
 - * Main
- * Testspielchen
- * Fragen und Anmerkungen

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;  
    }  
}
```

- * Was ist ggez?
 - * Features
 - * Betriebssysteme
 - * Benutzung
 - * Basic Template
- * Was ist RustSweeper?
 - * Erläuterung
 - * Spielregeln
- * **Implementierung**
 - *Felder
 - ***Mienen**
 - *Count
 - *EventHandler
 - *Main
- * Testspielchen
- * Fragen und
Anmerkungen

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;  
        }  
    }  
}
```

- * Was ist ggez?
 - * Features
 - * Betriebssysteme
 - * Benutzung
 - * Basic Template
- * Was ist RustSweeper?
 - * Erläuterung
 - * Spielregeln
- * **Implementierung**
 - *Felder
 - *Mienen
 - ***Count**
 - *EventHandler
 - *Main
- * Testspielchen
- * Fragen und Anmerkungen

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;  
    }  
}
```

- * Was ist ggez?
 - * Features
 - * Betriebssysteme
 - * Benutzung
 - * Basic Template
- * Was ist RustSweeper?
 - * Erläuterung
 - * Spielregeln
- * **Implementierung**
 - *Felder
 - *Mienen
 - ***Count**
 - *EventHandler
 - *Main
- * Testspielchen
- * Fragen und Anmerkungen

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;  
    }  
}
```

- * Was ist ggez?
 - * Features
 - * Betriebssysteme
 - * Benutzung
 - * Basic Template
- * Was ist RustSweeper?
 - * Erläuterung
 - * Spielregeln
- * **Implementierung**
 - *Felder
 - *Mienen
 - ***Count**
 - *EventHandler
 - *Main
- * Testspielchen
- * Fragen und Anmerkungen

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;  
}
```

- * Was ist ggez?
 - * Features
 - * Betriebssysteme
 - * Benutzung
 - * Basic Template
- * Was ist RustSweeper?
 - * Erläuterung
 - * Spielregeln
- * **Implementierung**
 - *Felder
 - *Mienen
 - ***Count**
 - *EventHandler
 - *Main
- * Testspielchen
- * Fragen und Anmerkungen

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;  
    }  
}
```

- * Was ist ggez?
 - * Features
 - * Betriebssysteme
 - * Benutzung
 - * Basic Template
- * Was ist RustSweeper?
 - * Erläuterung
 - * Spielregeln
- * **Implementierung**
 - *Felder
 - *Mienen
 - ***Count**
 - *EventHandler
 - *Main
- * Testspielchen
- * Fragen und Anmerkungen

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

- * Was ist ggez?
 - * Features
 - * Betriebssysteme
 - * Benutzung
 - * Basic Template
- * Was ist RustSweeper?
 - * Erläuterung
 - * Spielregeln
- * **Implementierung**
 - *Felder
 - *Mienen
 - *Count
 - ***EventHandler**
 - *Main
- * Testspielchen
- * Fragen und Anmerkungen

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())?;
        }
    }
}
```

- * Was ist ggez?
 - * Features
 - * Betriebssysteme
 - * Benutzung
 - * Basic Template
- * Was ist RustSweeper?
 - * Erläuterung
 - * Spielregeln
- * **Implementierung**
 - * Felder
 - * Mienen
 - * Count
 - * **EventHandler**
 - * Main
- * Testspielchen
- * Fragen und Anmerkungen

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

- * Was ist ggez?
 - * Features
 - * Betriebssysteme
 - * Benutzung
 - * Basic Template
- * Was ist RustSweeper?
 - * Erläuterung
 - * Spielregeln
- * **Implementierung**
 - *Felder
 - *Mienen
 - *Count
 - ***EventHandler**
 - *Main
- * Testspielchen
- * Fragen und Anmerkungen

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();  
}
```

- * Was ist ggez?
 - * Features
 - * Betriebssysteme
 - * Benutzung
 - * Basic Template
- * Was ist RustSweeper?
 - * Erläuterung
 - * Spielregeln
- * **Implementierung**
 - *Felder
 - *Mienen
 - *Count
 - *EventHandler
 - ***Main**
- * Testspielchen
- * Fragen und Anmerkungen

RustSweeper

Alexander Chmielus,
Jasmine Cavael

- * Was ist ggez?
 - * Features
 - * Betriebssysteme
 - * Benutzung
 - * Basic Template
- * Was ist RustSweeper?
 - * Erläuterung
 - * Spielregeln
- * Implementierung
 - *Felder
 - *Mienen
 - *Count
 - *EventHandler
 - *Main
- * **Testspielchen**
- * Fragen und
Anmerkungen

Testspielchen

Vielen Dank fürs Zuhören!

Habt ihr noch Fragen und/oder
Anmerkungen?

- * Was ist ggez?
 - * Features
 - * Betriebssysteme
 - * Benutzung
 - * Basic Template
- * Was ist RustSweeper?
 - * Erläuterung
 - * Spielregeln
- * Implementierung
 - *Felder
 - *Mienen
 - *Count
 - *EventHandler
 - *Main
- * Testspielchen
- * Fragen und
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

- * Was ist ggez?
 - * Features
 - * Betriebssysteme
 - * Benutzung
 - * Basic Template
- * Was ist RustSweeper?
 - * Erläuterung
 - * Spielregeln
- * Implementierung
 - *Felder
 - *Mienen
 - *Count
 - *EventHandler
 - *Main
- * Testspielchen
- * Fragen und Anmerkungen