Phaser:
States & Geometry

Paweł Łosek

Stany w Phaserze

- Metoda na podział logiki naszej gry na części (moduły),
- Każdy stan implementuje element cyklu życia gry (ładowanie, menu, właściwy ekran gry),
- Phaser udostępnia dwie klasy do zarządzania stanami: StateManager oraz State

State

- Reprezentuje pojedynczy stan gry,
- Posiada swój cykl życia (dalsze slajdy),
- Musi implementować co najmniej jedną z metod: preload, create, render lub update,
- Obiekt zadeklarowany jako Phaser.State “dziedziczy” domyślnie kilka ważnych pól będących obiektami frameworka (kolejny slajd),
- dokumentacja

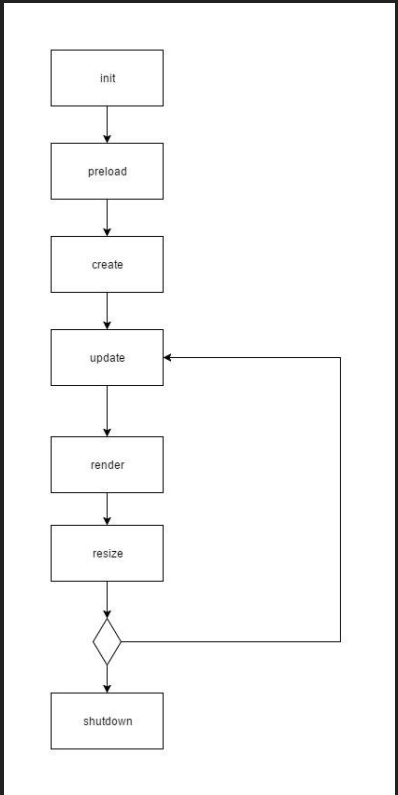
State - dziedziczone pola

Kilka przykładowych obiektów, do których bezpośredni dostęp ma obiekt stanu:

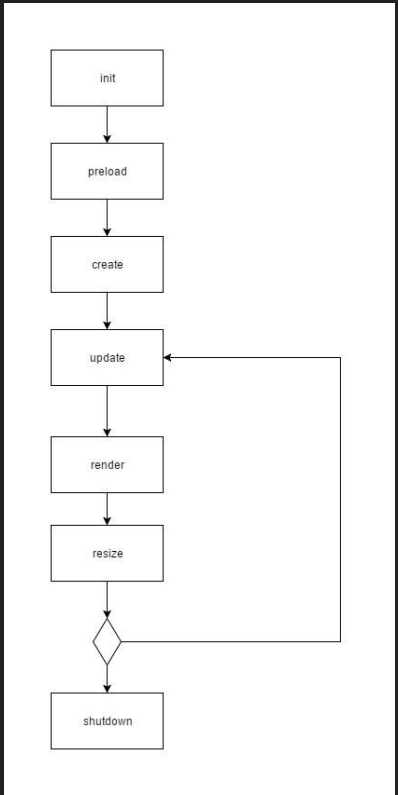
- Game - referencja do aktualnie używanego obiektu gry
- Cache - dostęp do obiektów załadowanych do pamięci podręcznej,
- Input - referencja do InputManagera,
- Math - referencja do zbioru użytecznych funkcji matematycznych,
- Time - zegar gry,
- Sound - manager dźwieku,
- Przykładowe odwołanie do pow. i szeregu innych obiektów:

this.game, this.physics etc.

Cykl życia stanu

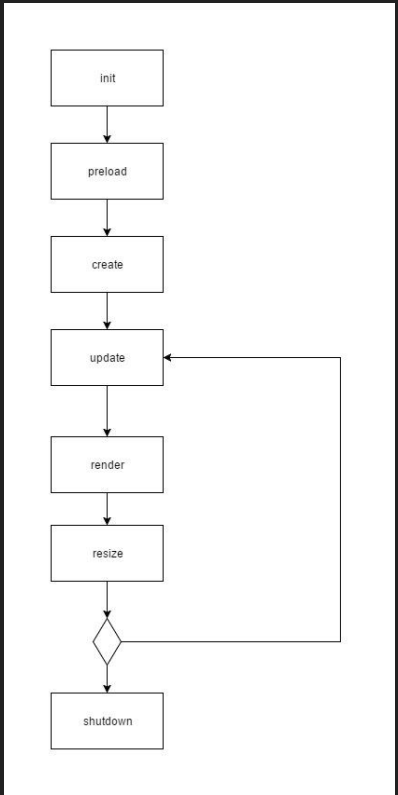


Cykl życia stanu



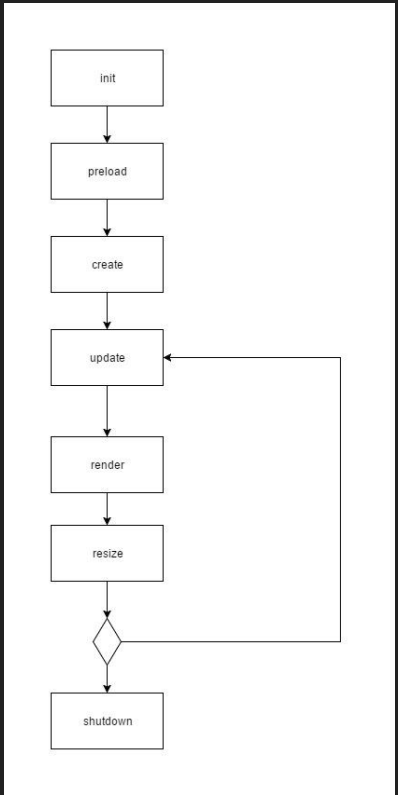
- init - w określonych przypadkach przekierowanie do innego stanu, inicjalizacja zmiennych potrzebnych do kolejnych faz

Cykl życia stanu



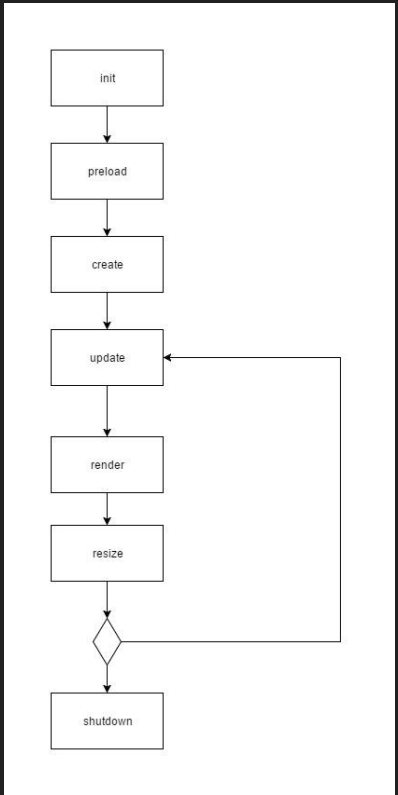
- preload - ładowanie obrazków i innych elementów potrzebnych do obsługi stanu gry,
- w trakcie tej fazy cyklu wywoływane są dwie dodatkowe metody:
  - loadUpdate - implementacja postępu ładowania (np. progressbar),
  - loadRender - nieużywana jeśli Phaser działa w trybie WebGL, powinna zawierać kod dot. logiki renderowania

Cykl życia stanu



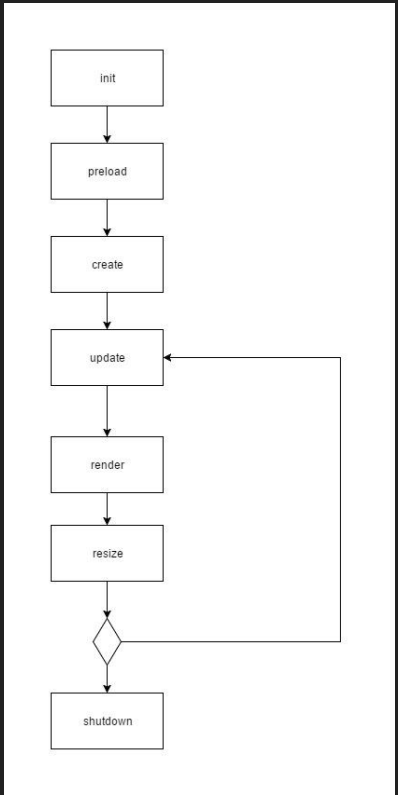
- create - bezpieczne stworzenie sprite’ów, cząsteczek oraz innych obiektów wykorzystywanych przez stan, korzystających z zasobów załadowanych w fazie preload.

Cykl życia stanu



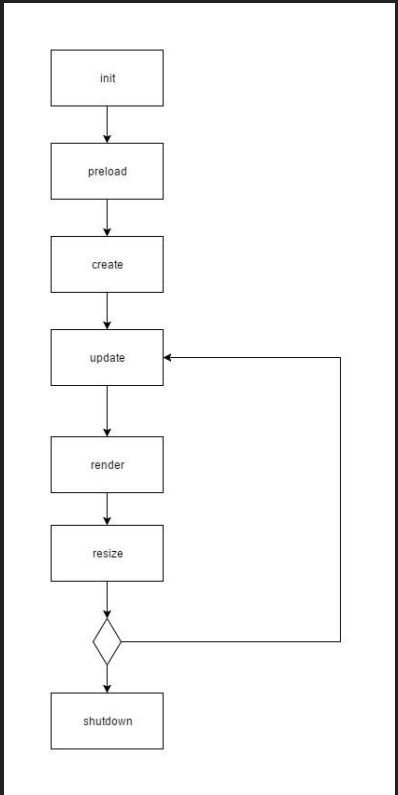
- update :
  - wywoływana przed każdym rysowaniem klatki
  - w przybliżeniu, 60 razy na sekundę (60 fps) dla komputera stacjonarnego,
  - główne obliczenia, takie jak obsługa inputu ze strony gracza (klawiatura, myszka etc.), sprawdzenie kolizji obiektów, inna logika

Cykl życia stanu



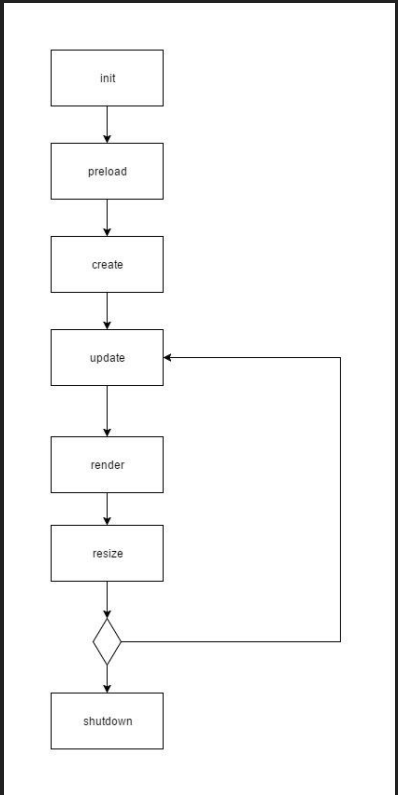
- render:
  - wywołana po odpowiadającej natywnej funkcji canvas/WebGL,
  - zazwyczaj implementuje się tutaj dodatkowe post-efekty
  - ewentualnie, nanosi się tutaj kształty ułatwiające debugowanie

Cykl życia stanu



- resize(opcjonalnie) - wywoływana przy zmianie rozmiaru kontenera gry (w trybie RESIZE)
  - pobiera dwa argumenty: nowa wysokość i szerokość kontenera,
  - umożliwia dostosowanie elementów responsywnych do nowego rozmiaru okna

Cykl życia stanu



- shutdown - wywoływana jako ostatnia metoda w cyklu życia stanu (np. przy przechodzeniu do kolejnego stanu)

Dodatkowe metody cyklu życia

- paused - wywoływane jeśli główna pętla gry zostanie zatrzymana (this.game.paused = true),
- pauseUpdate - wywoływana w miejsce update, jeżeli pętla gry jest wstrzymana,
- resumed - wywołana jeśli pętla gry zostaje wznowiona

StateManager

- Zarządzanie flow gry - stanami,
- Podstawowe funkcjonalności:
  - Metoda *add* (*game.state.add(key, state, autoStart)*)
    - key - unikalny identyfikator stanu,
    - state - definicja stanu (Phaser.State, obiekt JS lub funkcja),
    - autoStart - jeżeli true, wywołaj *start* bezpośrednio po dodaniu stanu (domyślnie false)
  - Metoda *start*
    - Powoduje przejście do stanu o identyfikatorze podanym jako argument,
    - Wywołuje funkcję *shutdown* aktualnego stanu
- dokumentacja

Link do repo

https://github.com/Nevaan/tipgk_lab

- Skonfigurowany projekt Phaser,
- Kilka zdefiniowanych stanów,
- Jeden ze sposobów implementacji,
- Plik do uruchomienia: index.html

Geometria, kształty w Phaser

- Phaser posiada funkcjonalność rysowania prymitywnych kształtów
  - Łuki (arc),
  - Krzywe Beziera (bezierCurveTo),
  - Koła (drawCircle),
  - Elipsy (drawEllipse),
  - Wielokąty (drawPolygon),
  - Prostokąty (drawRect),
  - Prostokąty o zaokrąglonych wierzchołkach (drawRoundedRect),
  - Proste (lineTo)

Geometria, kształty w Phaser - howto

1. Stworzenie obiektu Phaser.Graphics

```
var circle = this.add.graphics(this.game.world.centerX, this.game.world.centerY)
```

2. Wywołanie na nim metody *beginFill()*

```
circle.beginFill(0x434986);
```

3. Rysowanie właściwego obiektu

```
circle.drawCircle(x,y,d);
```

4. Wyrysowanie kształtu

```
circle.endFill();
```

Geometria, kształty w Phaser - howto - Polygon

1. Wielokąt definiowany jest jako zbiór punktów
2. Korzysta z definicji Phaser.Point
3. Metoda drawPolygon przyjmuje zbiór punktów (tablicę) jako argument

```
var points = [ new Phaser.Point(200, 100), new Phaser.Point(350, 100), new Phaser.Point(375, 200), new Phaser.Point(150, 200) ]
```

4. Wywołanie metody

```
polygon.drawPolygon(points);
```

Geometria, kształty w Phaser - dodatkowe metody

- *lineStyle(lineWidth,color,alpha)* - styl obramowania
- *moveTo(x,y)* - przesunięcie aktualnego punktu wyznaczającego miejsce rysowania,
- *clear()* - czyszczenie grafiki
- dokumentacja

Czas na testowanie

Dzięki za uwagę