SFML - *Sprite* i tekstura

Objektno programiranje - 10. vježbe (2. dio)

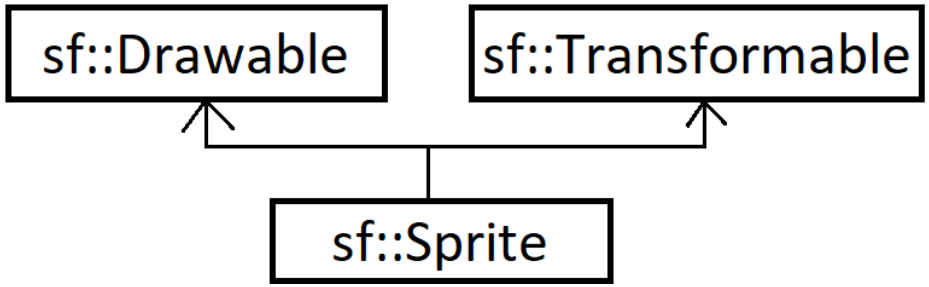
dr. sc. Sebastijan Horvat

Prirodoslovno-matematički fakultet,
Sveučilište u Zagrebu

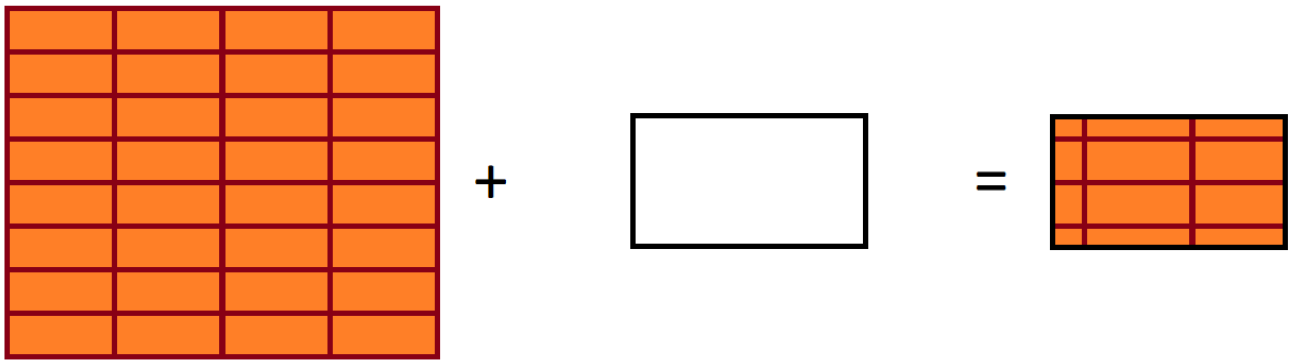
22. svibnja 2024. godine

Sprite

Dijagram nasljeđivanja:



- *sprite* je objekt koji možemo crtati i transformirati (povećati/smanjiti, pomaknuti, rotirati)
- no, sam po sebi nema nikakvih grafičkih podataka - nešto što bi iscrtali - zato trebamo teksturu



tekstura + pravokutnik = *sprite*

Tekstura

- slika (niz piksela) na grafičkoj kartici koju koristimo pri crtanju
- nalazi se na grafičkoj kartici (za razliku od obične slike) $\Rightarrow$ brzo crtanje
- može se pretvoriti u/dobiti iz `sf::Image` (to je spora operacija zbog prijenosa između grafičke kartice i glavne memorije)
- pikseli `sf::Image` nalaze se u memoriji sustava (zato operacije na njima najbrže moguće), a pikseli teksture u video memoriji (zato spori za dobivanje i mijenjanje, ali brzi za crtanje)

`sf::Sprite` objekti imaju referencu na teksturu, ne njenu kopiju

$\Rightarrow$ ne uništiti teksturu dok ju neki `sf::Sprite` objekt koristi

Primjer greške (vezano uz prethodni slajd)

Primjer.

```
sf::Sprite StvoriSprite(std::string putanja){
    sf::Texture tekstura;
    tekstura.loadFromFile(putanja);
    ...
    return sf::Sprite(tekstura);
}
```

`sf::Texture` i `sf::Sprite` konstruktori

`sf::Texture::Texture()`

- *defaultni* konstruktor - stvara praznu teksturu

`sf::Texture::Texture(const Texture& copy)`

- *copy* konstruktor

`sf::Sprite::Sprite()`

- *defaultni* konstruktor - stvara prazan *sprite* bez teksture

`sf::Sprite::Sprite(const Texture& texture)`

- konstruira *sprite* sa zadanom teksturom

`sf::Sprite::Sprite(const Texture& texture, const IntRect& rectangle)`

- stvara *sprite* za zadani pravokutni dio teksture

Učitavanje teksture iz datoteke

- podržani formati slika: bmp, png, tga, jpg, gif, psd, hdr, pic

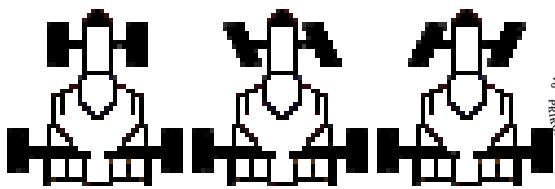
```
bool sf::Texture::loadFromFile(const std::string&
    filename, const IntRect& area = IntRect())
```

- vraća `true` ako datoteka uspješno učitana (inače `false` - u tom slučaju tekstura nije promijenjena)
- `filename` je putanja do slike koju učitavamo
- `area` je dio slike koji želimo učitati (po *defaultu* je prazan `IntRect` - tada se učita sve)
- ako `area` pravokutnik prelazi granice slike, prilagođava se kako bi odgovarao dijelu slike
- maksimalna veličina teksture koja se može koristiti može se saznati pomoću `getMaximumSize` funkcije

Primjer.

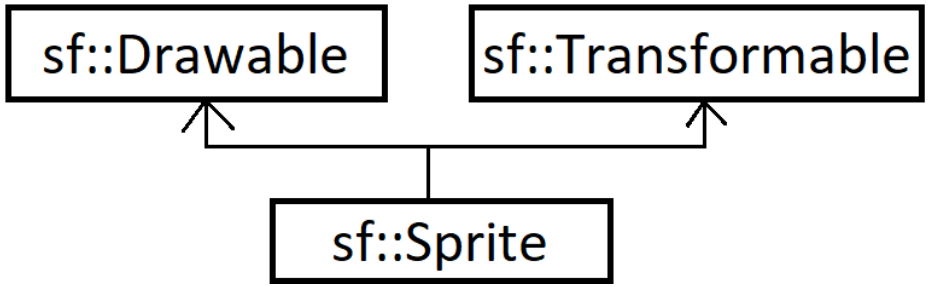
```
int main() {
    sf::RenderWindow window(sf::VideoMode(640,
            480), "Prozor!");
    sf::Texture tekstura;
    tekstura.loadFromFile("formule.png");
    sf::Sprite formula(tekstura);
    while (window.isOpen()) {
        ...
        window.clear(sf::Color::White);
        window.draw(formula);
        window.display();
    }
    return 0;
    }
```

Slika "formule.png" (u istoj mapi kao i naš projekt):



Objašnjenje - zašto možemo koristiti `draw`

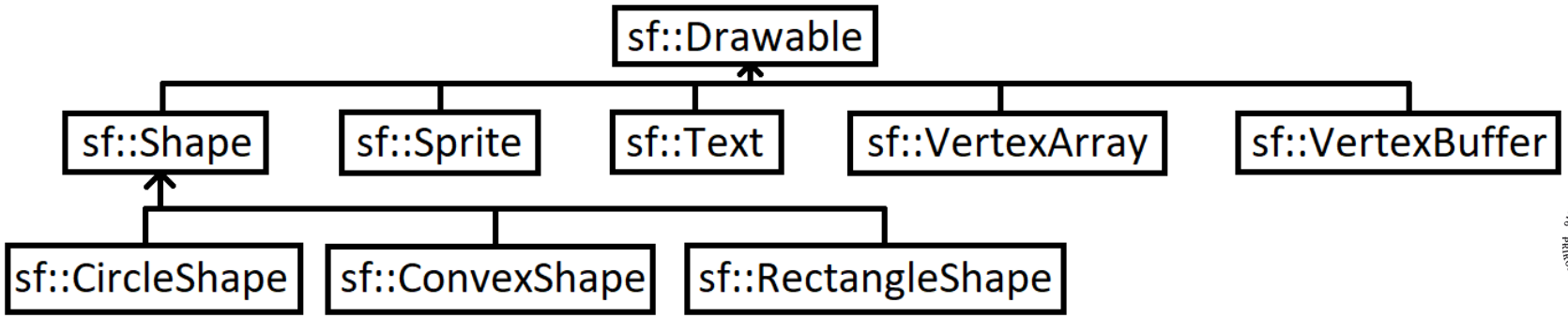
- podsjetnik na dijagram nasljeđivanja za `sf::Sprite`:



- klasa `sf::RenderWindow` nasljeđuje `sf::RenderTarget` klasu koja ima funkciju članicu:

```
void sf::RenderTarget::draw(const Drawable& drawable,
  const RenderStates& states = RenderStates::Default)
```

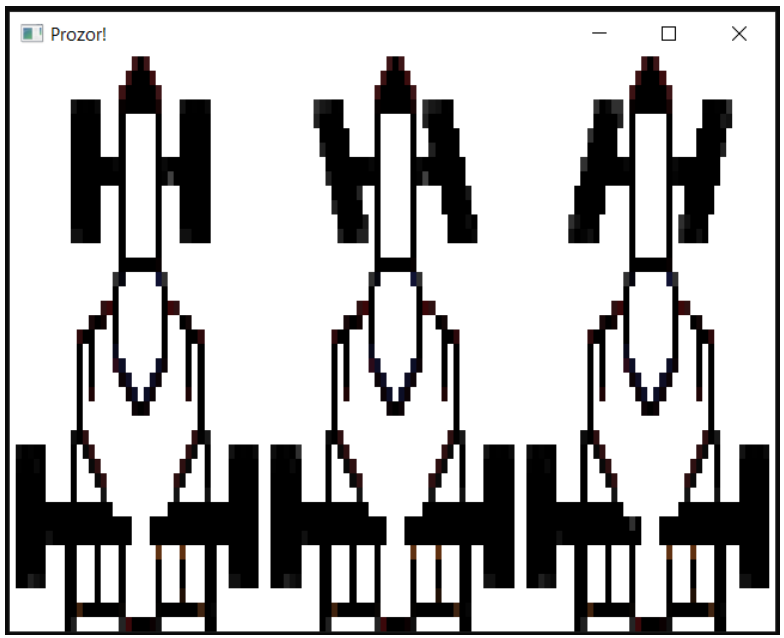
- dijagram nasljeđivanja za `sf::Drawable`



Postavljanje *spritea* preko cijelog prozora

```
...
sf::Sprite formula(tekstura);
formula.setScale(float(window.getSize().x) /
   tekstura.getSize().x,
   float(window.getSize().y)/tekstura.getSize().y);
...
```

Dobiven prikaz prozora:

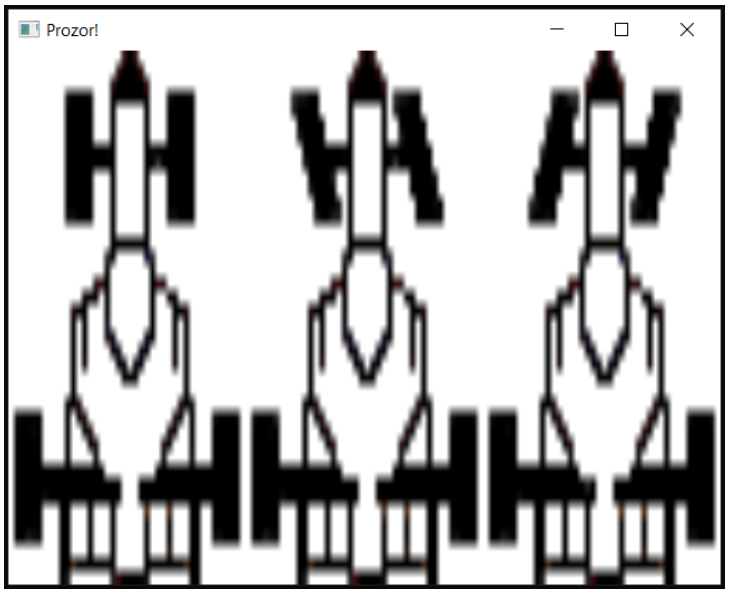


Glatki prikaz teksture

- prethodna tekstura sad dosta uvećana
- možemo piksele oko granice lika staviti manje vidljivima - postiže se zamućivanjem slike

```
...
tekstura.loadFromFile("formule.png");
tekstura.setSmooth(true);
...
```

Dobiven prikaz prozora (usporedite s prikazom s prethodnog slajda):

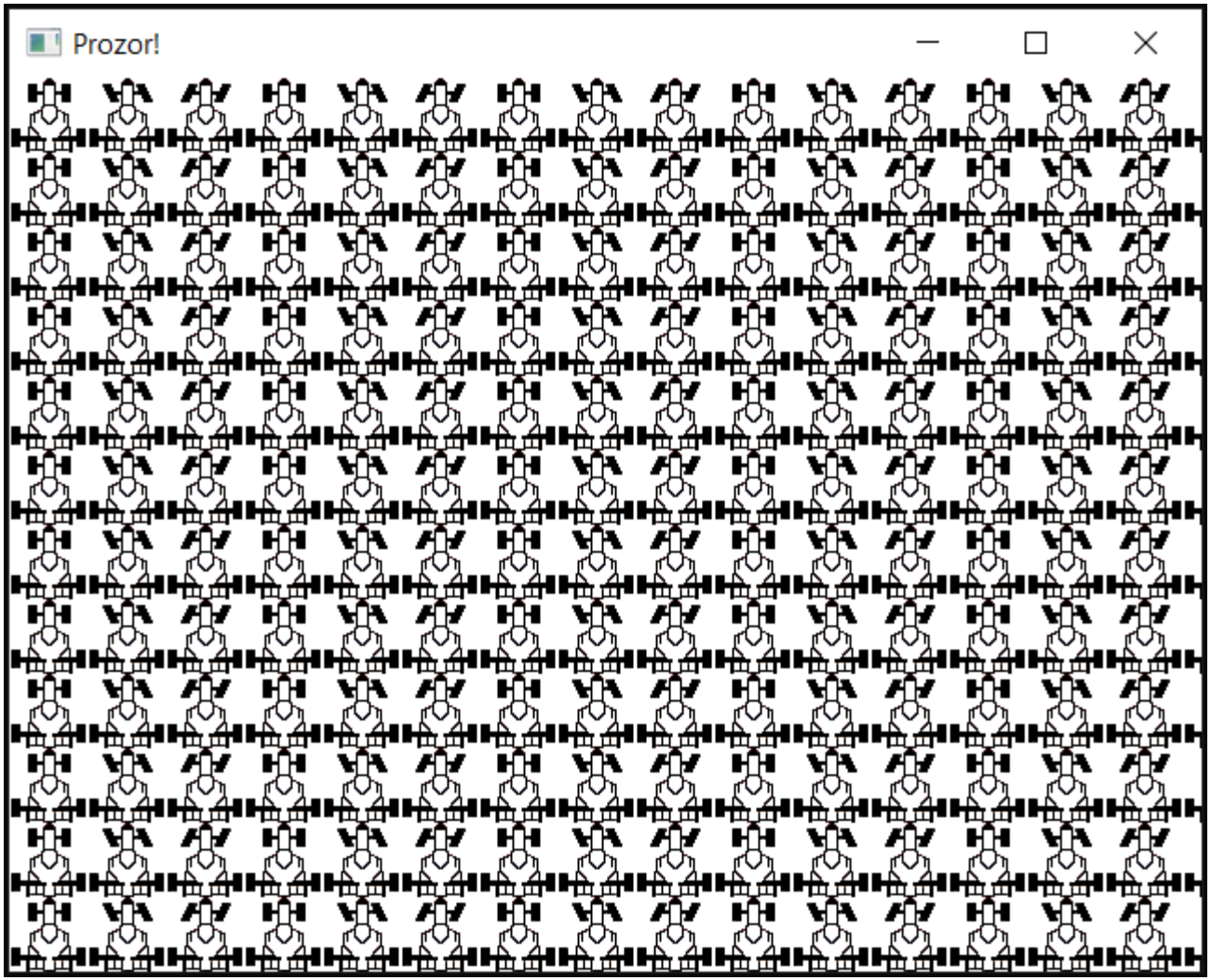


Ponavljanje teksture

- možemo teksturu ponavljati unutar područja *spritea*
- to radi samo ako *sprite* prikazuje pravokutnik koji je veći od teksture (kojom ćemo taj pravokutnik popločiti)
- u donjem primjeru pravokutnik koji prikazuje *sprite* odgovara području prozora:

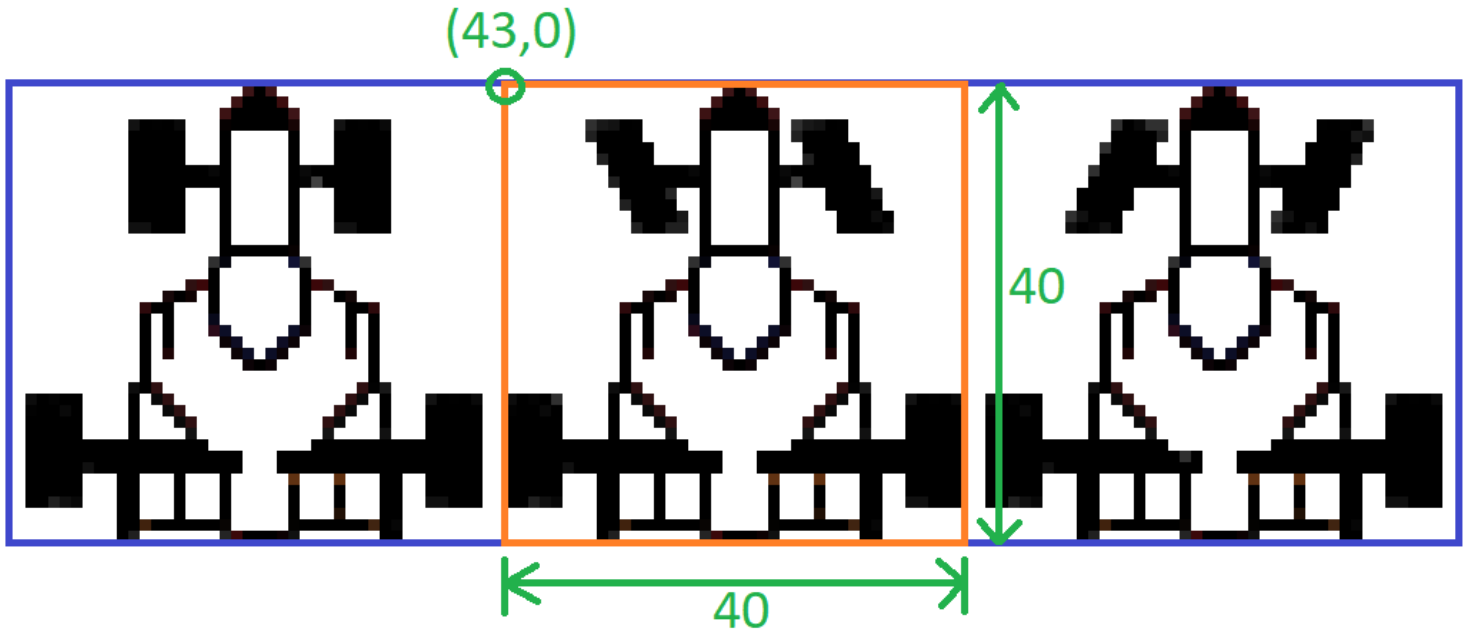
```
...
tekstura.loadFromFile("formule.png");
tekstura.setRepeated(true);
sf::Sprite formula(tekstura);
formula.setTextureRect(sf::IntRect(0,0,
    window.getSize().x,window.getSize().y));
...
```

Prikaz dobiven kodom s prethodnog slajda

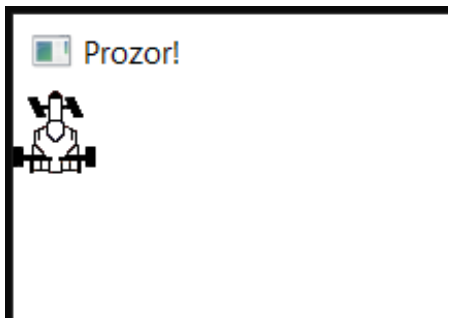


Prikaz dijela teksture

- prethodno prikazanu funkciju `setTextureRect` možemo iskoristiti kako bi prikazali samo srednju formulu s naše slike




```
formula.setTextureRect(sf::IntRect(43,0,40,40));
```



Promjena boje *spritea*

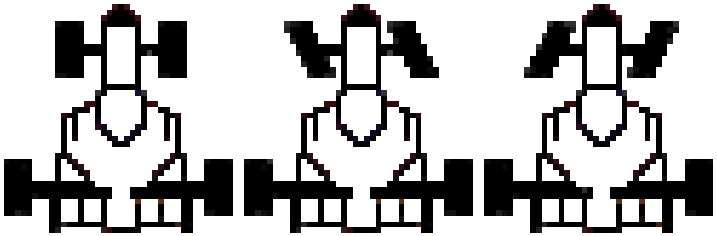
- boja je modulirana (pomnožena) s teksturom *spritea*
- također se tako može mijenjati transparentnost *spritea*

```
sf::Texture tekstura;
tekstura.loadFromFile("formula.png");
sf::Sprite formula(tekstura);
formula.setColor(sf::Color::Red);
```

Napomena. U gornjem primjeru korištene su redom boje: `sf::Color::Red` (kao u prikazanom kodu), `sf::Color::Green` i `sf::Color::Blue` te je korištena slika iz prošle prezentacije.

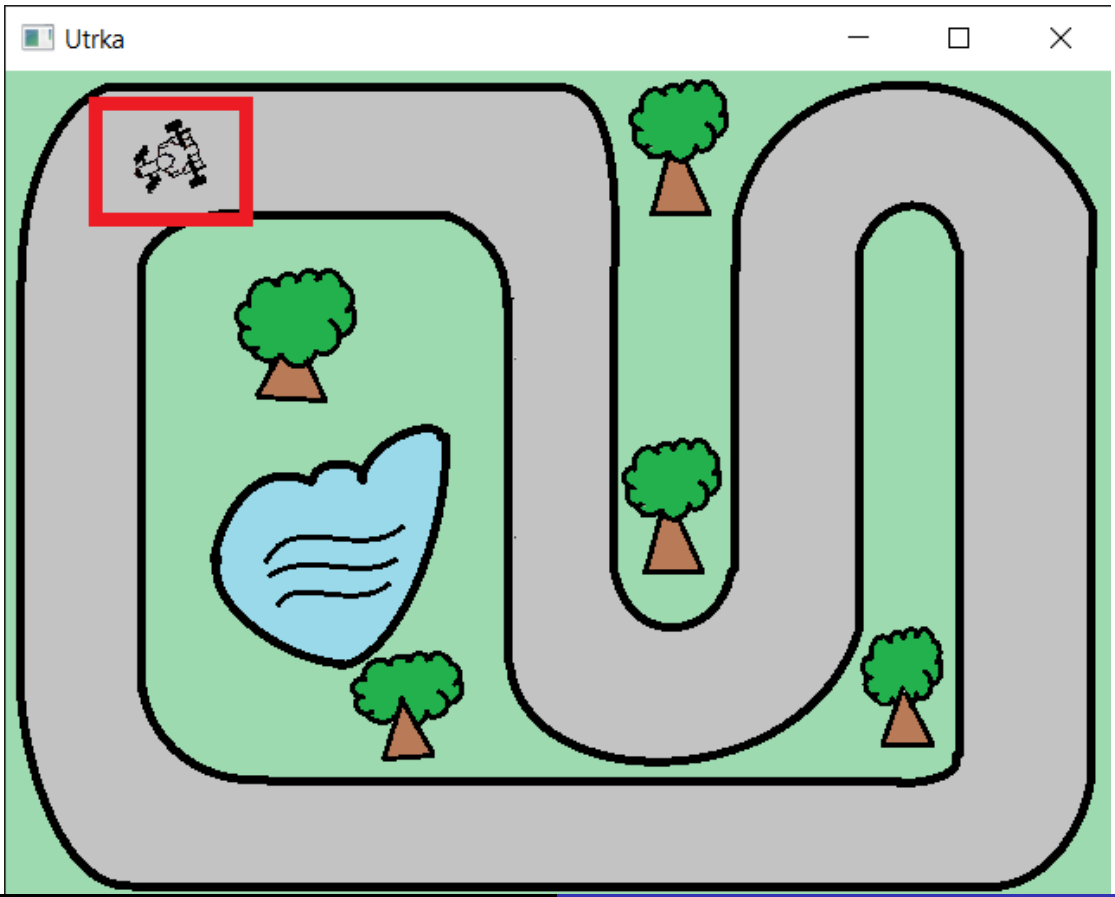
Napomena: Više *spriteova*, ali manje tekstura

- poželjno je korištenje što manje tekstura
- razlog: promjena teksture je skupa operacija za grafičku kartice
- najbolje performanse stoga daje crtenje više *spriteova* koji koriste istu teksturu (spritesheet)



Zadatak

Zadatak. Dopuniti kod iz prošle prezentacije tako da formula ima ispravno okrenute prednje kotače ovisno o tome kreće li se naprijed/nazad ili lijevo ili desno (koristeći `"formule.png"` datoteku).



Rješenje

```
class Igra {
 ...
 private:
  ...
  int pozT = 0; //x-koordinata lijevog ruba
                //dijela teksture za prikaz
};

Igra::Igra() :  p(...)  {
  tekstura.loadFromFile("formule.png");
  sprite.setTexture(tekstura);
  sprite.setTextureRect(sf::IntRect(pozT,0,40,40));
  ...
}
```

Rješenje (nastavak)

```
void Igra::obradiUlaz() {
  pozT = 0;
  ...
}

if(sf::Keyboard::isKeyPressed(sf::Keyboard::Left)){
  pozT = (smjer == 1) ?  43 : 85;
  ...
  }

if(sf::Keyboard::isKeyPressed(sf::Keyboard::Right)){
  pozT = (smjer == 1) ?  85 : 43;
  ...
  }
}
void Igra::renderiraj() {
  ...
  sprite.setTextureRect(sf::IntRect(pozT,0,40,40));
  p.crtaj(sprite);
  ...
}
```