SFML - *Sprite* i tekstura

Objektno programiranje - 7. vježbe (1. dio)

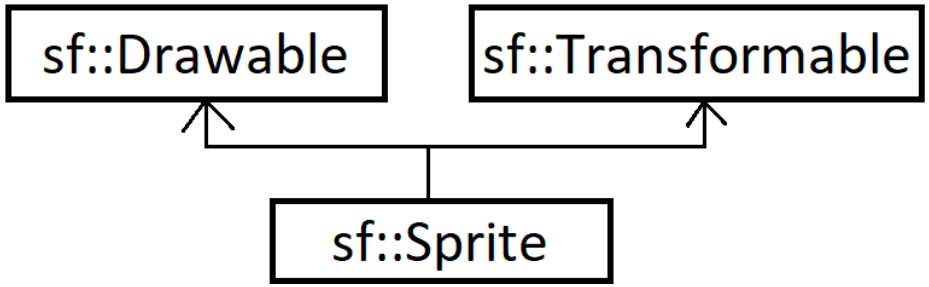
Marko Živković

Prirodoslovno-matematički fakultet,
Sveučilište u Zagrebu

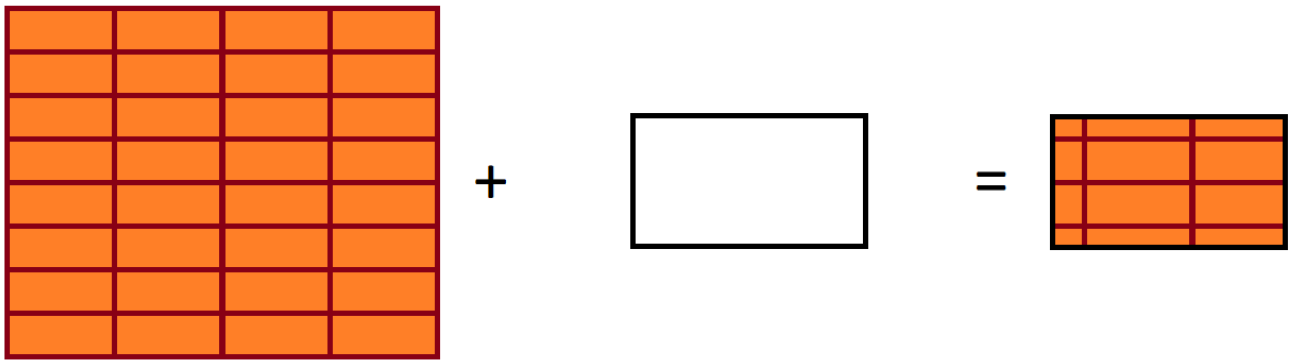
14. travnja 2026. godine

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)

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

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

`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)`

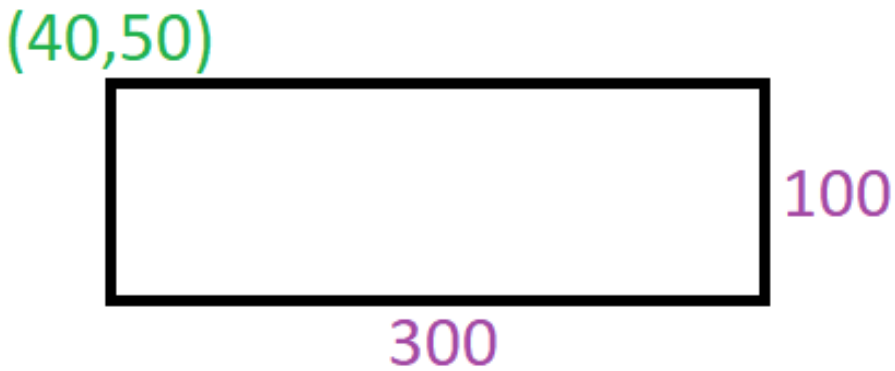
- stvara *sprite* za zadani pravokutni dio teksture

Pravokutnici: `Rect` predložak klase

- Služi za manipulaciju pravokutnika sa stranicama koje su paralelne osima - takav je definiran **gornjim lijevim kutem** i **veličinom**.
- Uz *defaultni* konstruktor (`sf::Rect<T>::Rect()`), imamo i konstruktor koji prima 4 podatka koji određuju takav pravokutnik:

```
sf::Rect<T>::Rect(T rectLeft, T rectTop,
                  T rectWidth, T rectHeight)
```

Primjer. `sf::Rect<int> p(40,50,300,100);`



`sf::Rect<int> p;` isto je kao `sf::Rect<int> p(0,0,0,0);`

Predložak klase `Rect`

- zbog jednostavnosti, SFML koristi sljedeće dvije instance (preuzeto iz datoteke `Rect.hpp` - link):

```
typedef Rect<int>    IntRect;
typedef Rect<float>  FloatRect;
```

- `Rect` predložak klase definiran iz praktičnih razloga pa ima sve dijelove javne $\Rightarrow$ možemo izravno koristiti **`left`**, **`top`**, **`width`** i **`height`** varijable članice (nisu funkcije pa ne pišemo () iza!)

Primjer.

```
sf::IntRect p(20, 30, 10, 50);
cout << p.left  << "," << p.top    << ","
     << p.width << "," << p.height << endl;
```

Ispis: `20,30,10,50`

Učitavanje teksture iz datoteke

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

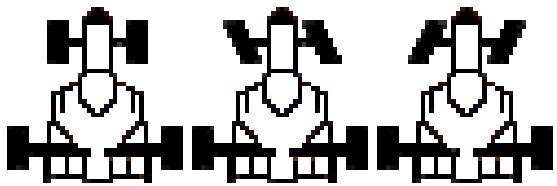
```
bool sf::Texture::loadFromFile(const std::string&
    filename, const IntRect& area = IntRect())
```

- 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

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;
    }
```

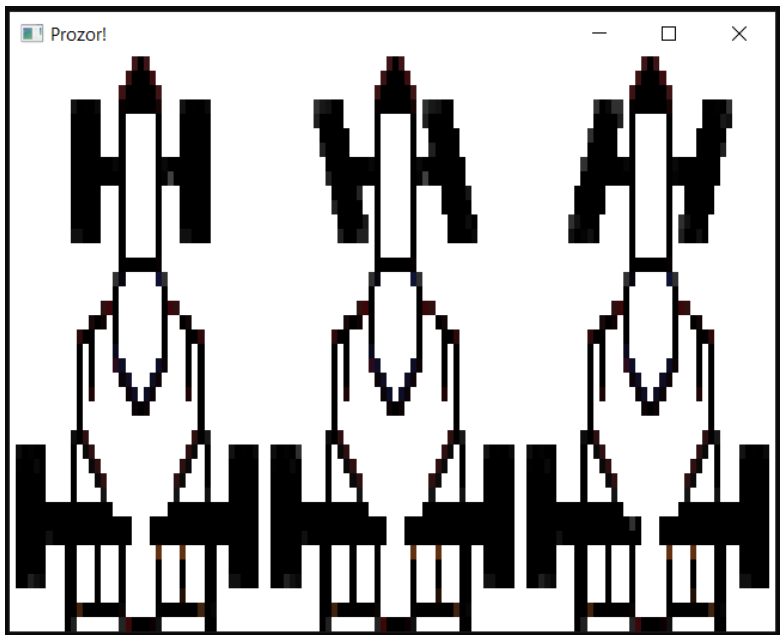
Slika "formule.png" (u istoj mapi kao i naš projekt):



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:

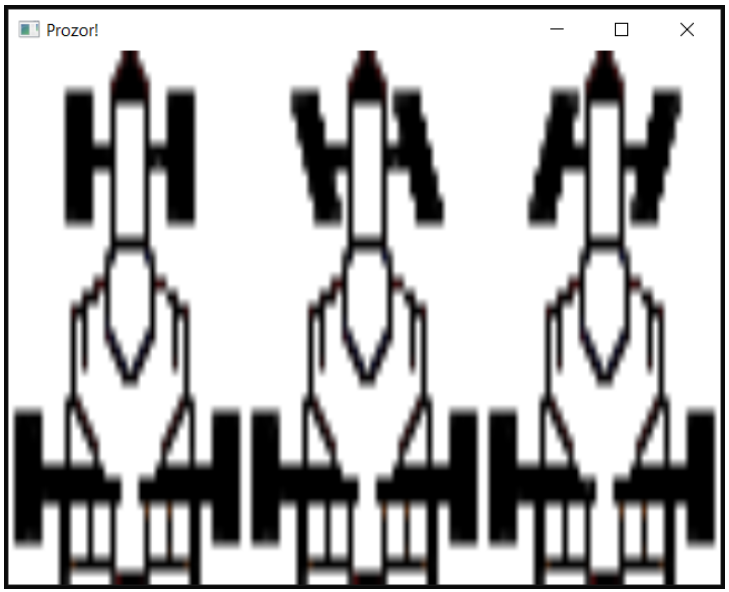


Glatki prikaz teksture

- 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):

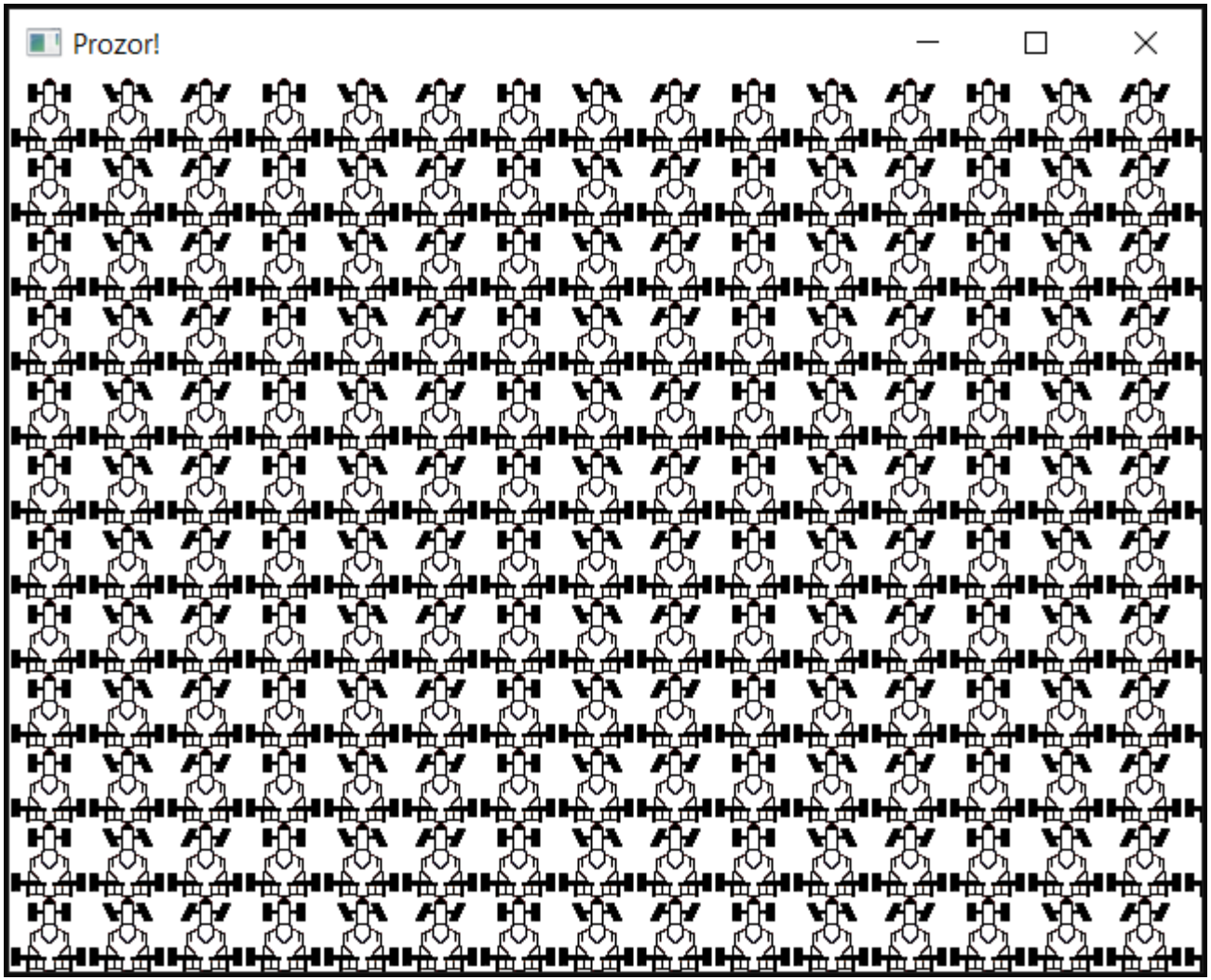


Ponavljanje teksture

- 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:

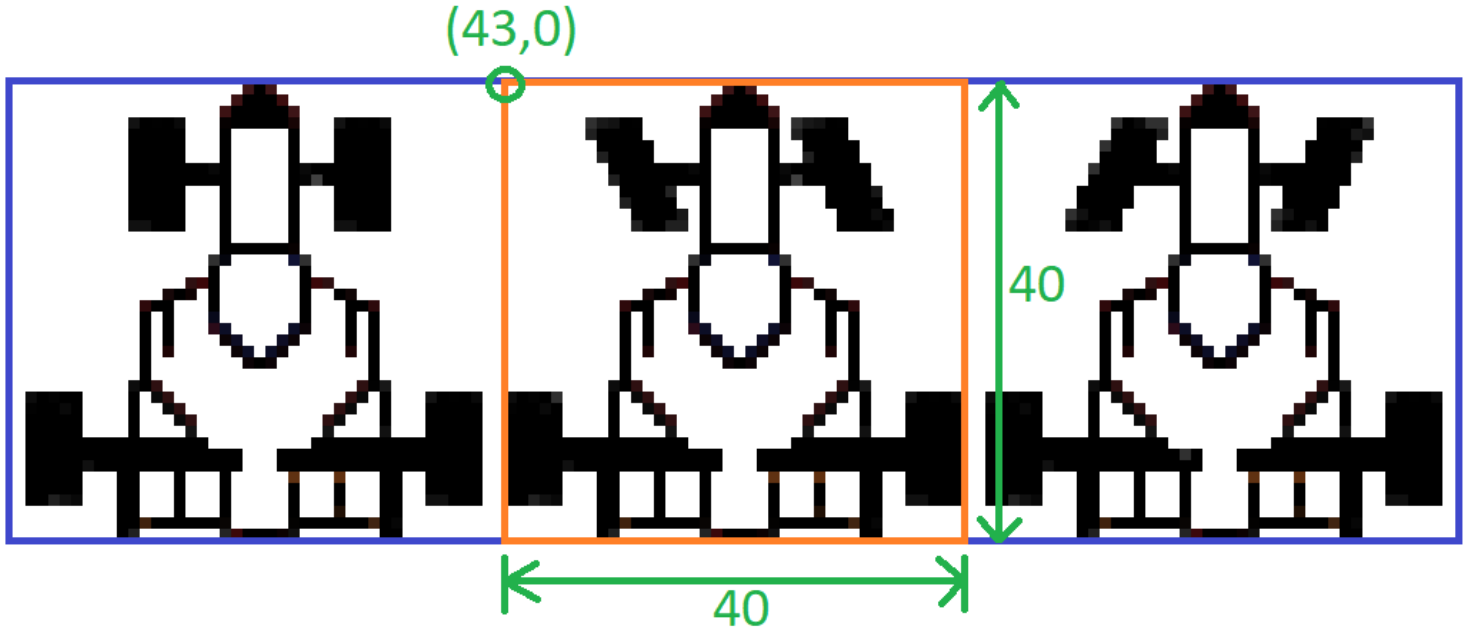
```
...
tekstura.loadFromFile("formule.png");
tekstura.setRepeated(true);
sf::Sprite formula(tekstura);
formula.setTextureRect(sf::IntRect(0,0,
    window.getSize().x,window.getSize().y));
...
```

Prikaz dobiven kodom s prethodnog slajda

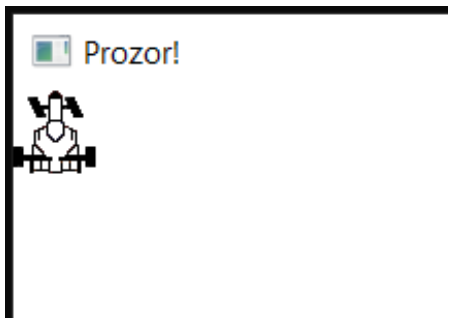


Prikaz dijela teksture

- 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*

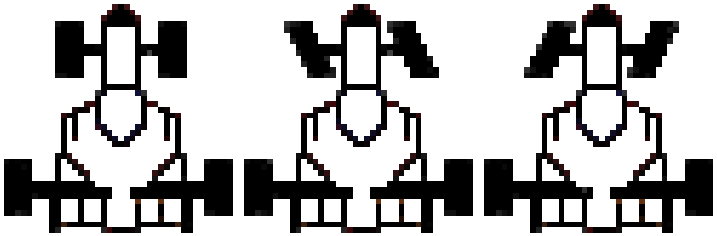
- boja je modulirana (pomnožena) s teksturom *spritea*
- također se tako može mijenjati transparentnost *spritea*

```
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.

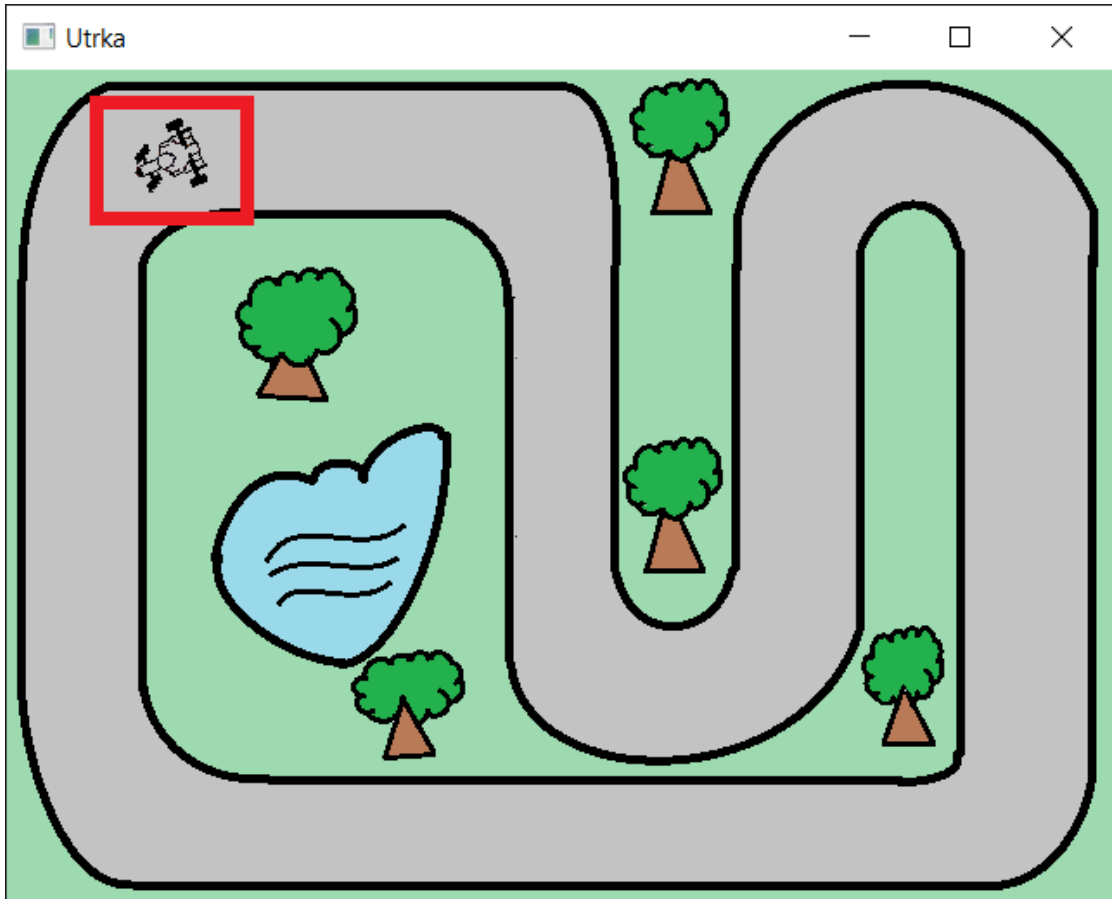
Napomena: Više *spriteova*, ali manje tekstura

- 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 1. Dopunite kod iz prošle prezentacije tako da formula ima ispravno okrenute prednje kotače ovisno o tome okreće li se lijevo ili desno (koristeći `"formule.png"` datoteku).



Rješenje

```
class Igra {
 ...
 private:
  ...
  int pozT = 1; //x-koordinata lijevog ruba
                //dijela teksture za prikaz
};

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

Rješenje (nastavak)

```
void Igra::innerUpdate() {
  pozT = 1;
  ...
}

if(sf::Keyboard::isKeyPressed(sf::Keyboard::Left)){
  pozT = 43;
  ...
  }

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

Zadatak

Zadatak 2. Dopunite kod tako da formula ispravno skreće samo ako ide naprijed ili nazad.

Rješenje

```
if(sf::Keyboard::isKeyPressed(sf::Keyboard::Left)){
  pozT = 43;
  if (sf::Keyboard::isKeyPressed(sf::Keyboard::Up))
    t.setRotation(kut -= brzina_okreta*deltaTime);
  if
  (sf::Keyboard::isKeyPressed(sf::Keyboard::Down))
    t.setRotation(kut += brzina_okreta*deltaTime);
}

if(sf::Keyboard::isKeyPressed(sf::Keyboard::Right)){
  pozT = 85;
  ...
  }
}
```