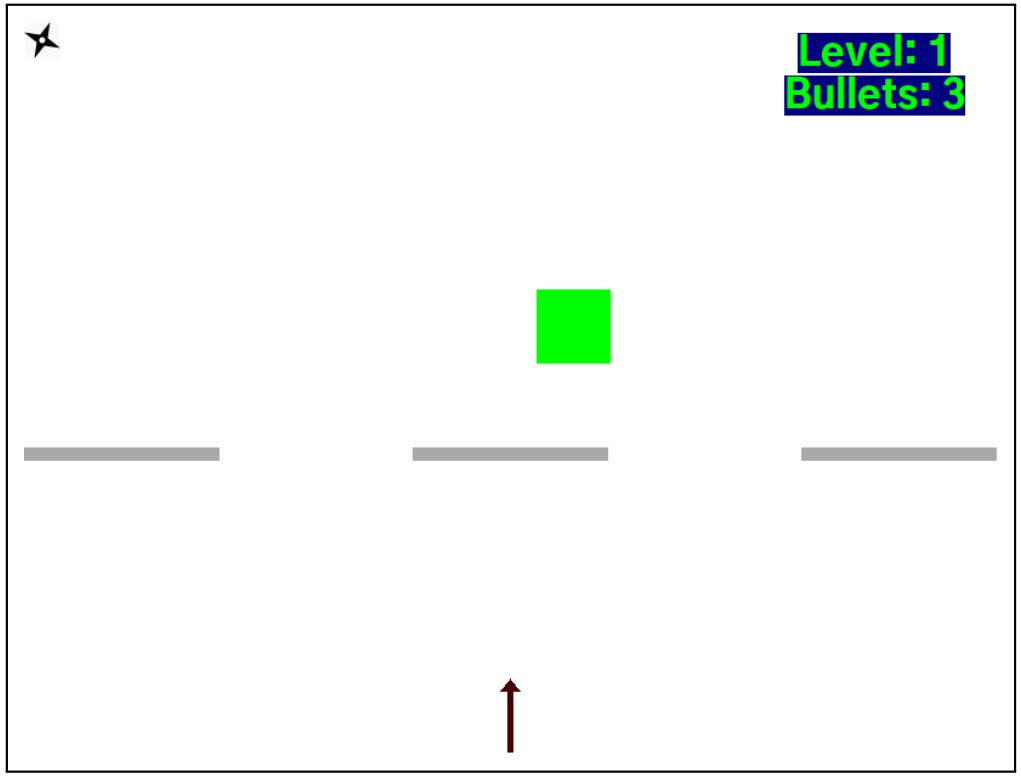
Gađanje mete

Vasilić Marko IN/2019
Savković Jovan IN/2019

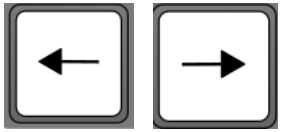
Opis problema

2D igrica - gađanje mete
Igrač bira projektil kojim gađa metu, ali na polju se nalaze prepreke u vidu debelih i tankih zidova. Igrač mora da izabere projektil koji će najbolje obaviti posao. Šuriken će uništiti tanke zidove, lopta će se odbiti od debele zidove. Treba kombinovati projektile tako da se oslobodi put do mete.



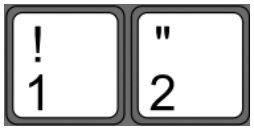
Kontrole

- Ciljanje:



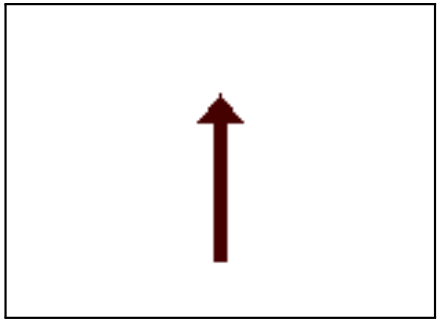
- Pucanje:

- Biranje municije:



- Resetovanje igrice:

- Zaustavljanje aktivnog projektila:



Strelica za ciljanje

Elementi rešenja

- Kinematika - pravolinijsko kretanje i rotaciono kretanje
  Rešenje - RK4
- Detekcija kolizije
  Rešenje - GJK

Kinematika

Pravolinijsko i rotaciono kretanje je pretstavljeno kao diferencijalna jednačina drugog reda.
Pomoću RK4 algoritma se aproksimira vrednost datih funkcija kretanja pri svakom koraku igrice.

```
def rk4It(xi, yi, h, f):
    k1 = f(xi, yi)
    k2 = f(xi + h / 2, yi + 0.5 * k1 * h)
    k3 = f(xi + h / 2, yi + 0.5 * k2 * h)
    k4 = f(xi + h, yi + k3 * h)
    return yi + h / 6 * (k1 + 2 * k2 + 2 * k3 + k4)
```

Detekcija kolizije

Detekcija kolizije funkcioniše za bilo koju kombinaciju krugova, poligona i duži

```
def gjk(s1, s2):
    if s1[0] == s2[0]:
        return True
    d = pygame.Vector2(-1, -1)
    simplex = [support_point(s1, s2, d)]
    d = ORIGIN - simplex[0]
    for i in range(100):
        A = support_point(s1, s2, d)
        if A.dot(d) < 0:
            return False
        simplex.append(A)
        if handleSimplex(simplex, d):
            return True
```

Detekcija kolizije - šuriken

Šuriken nije konveksan poligon, tako da gjk ne bi radio za njega. Ovaj problem je rešen tako što se šuriken posmatra kao skup 4 konveksna poligona. U koliko se detektuje kolizija sa bilo kojim od poligona koji ga sačinjavaju, smatra se da je šuriken došao u kontakt sa nečim.

```
for i in range(0, 4):
    self.blades.append([self.center + self.blade_from_center[0].rotate_rad(self.angle + i * (np.pi / 2)),
                        self.center + self.blade_from_center[1].rotate_rad(self.angle + i * (np.pi / 2)),
                        self.center + self.blade_from_center[2].rotate_rad(self.angle + i * (np.pi / 2)),
                        self.center + self.blade_from_center[3].rotate_rad(self.angle + i * (np.pi / 2))
                        ])
```

Odbijanje o zidove

Projektil "lopta" će se odbiti od zidove.

```
def wall_ball_collision(e1, e2):
    if isinstance(e1, entities.Ball):
        point = point_of_collision(e1, e2)
        new_way(e1, point)
    else:
        point = point_of_collision(e2, e1)
        new_way(e2, point)
```

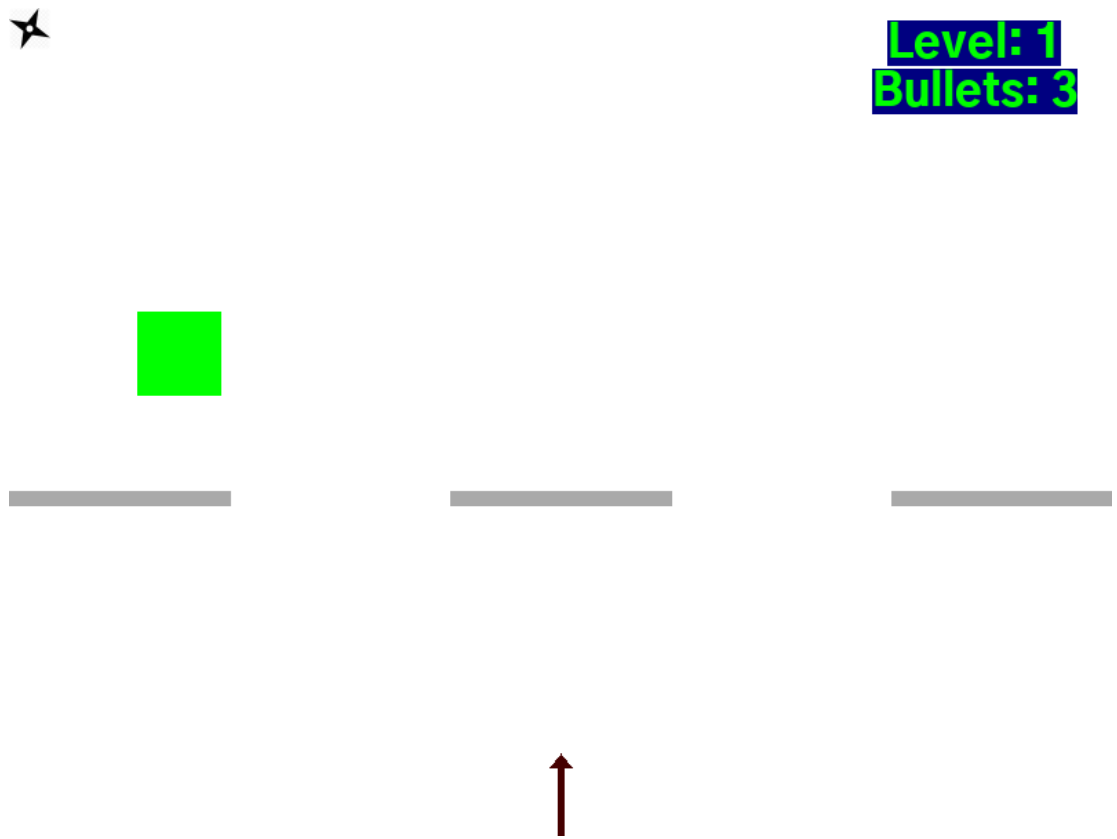
```
def new_way(ball, point):
    CKA = point - ball.get_center()
    angle = CKA.angle_to(ball.velocity)
    if abs(angle) < 90:
        new_velocity = -ball.velocity.rotate(-angle * 2)
        ball.velocity[0] = new_velocity[0]
        ball.velocity[1] = new_velocity[1]
```

Nivoi

Igrica se sastoji od 3 novoa koji služe da demonstriraju funkcionalnosti igre.
Za svaki nivo igrač ima na raspolaganju 3 projektila da pogodi metu, u koliko ne pogodi kreće ispočetka.

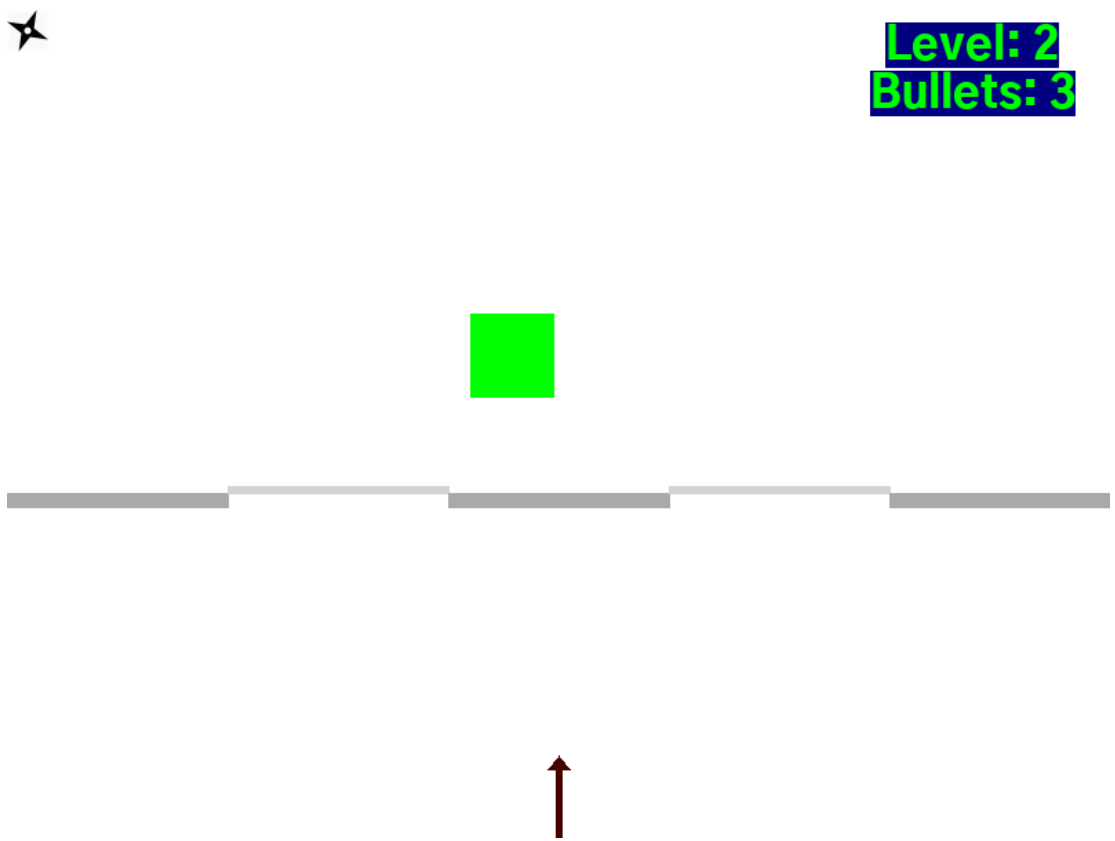
Nivo 1

Jednostavno gađanje mete.



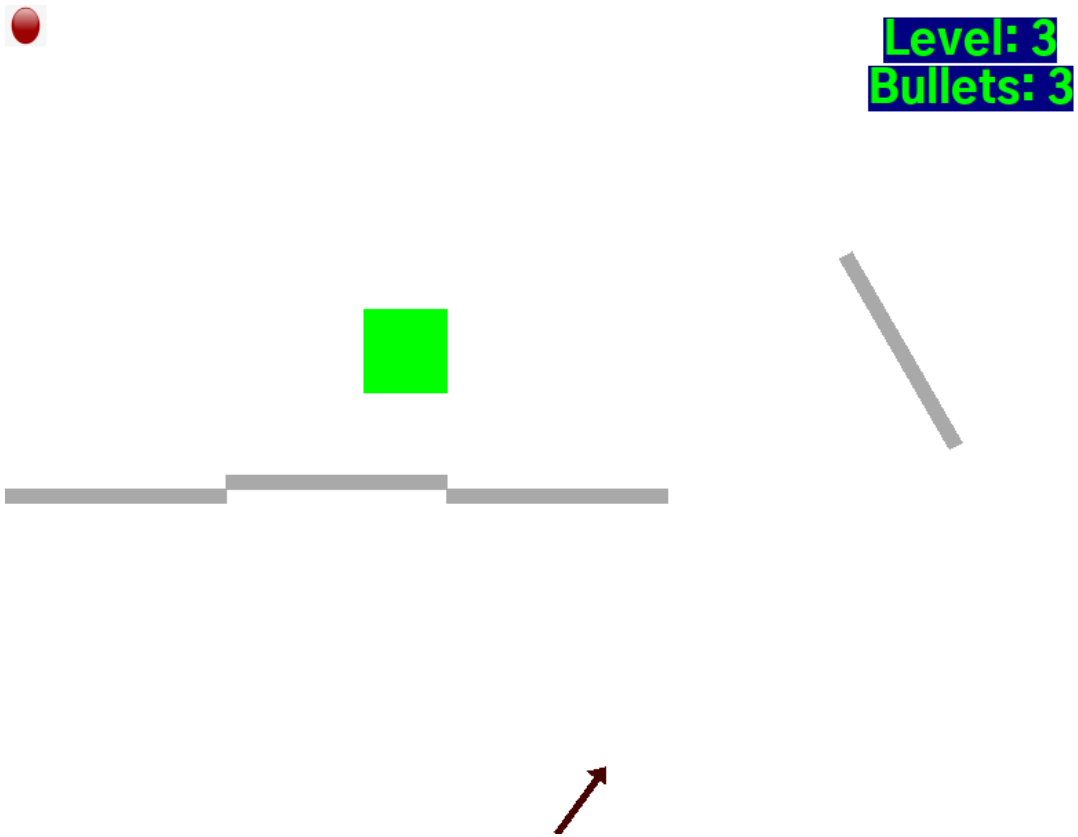
Nivo 2

Igrač mora prvo da pomoću šurikena uništi jedan od tankih zidova, zatim da pogodi metu proizvoljnim projektilom.



Nivo 3

Igrač mora da odbije lopticu od dijagonalni zid na desnoj strani ekrana da bi pogodio metu.



Hvala na pažnji