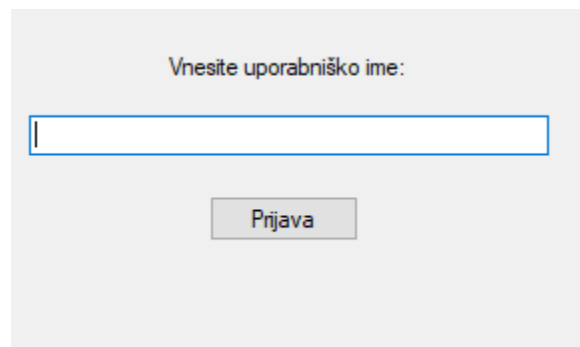


Požrešna kačica

Za projekt pri predmetu programiranje 3 sem si izbral igrico požrešna kačica. V tej igri vedno, ko kača poje sadež, se poveča svoj telo in nov sadež se pojavi. Če se kača zaleti v steno ali samo vase se igra zaključi. Cilj igre je da igralec pobere čim več sadežev in zbere točk preden se usodno zaleti.

To je bila osnovna igra, ki sem jo zasledil na internetu in YouTube. Tako da, bilo se je potrebno spomniti nekaj dodatnih in novih stvari, ki bodo to igrico povzpele na višjo raven.

Ob zagonu igre dobimo oknu, ki zahteva vpis uporabniškega imena. Ta bo kasneje uporabljen za beleženje točk na lestvici.

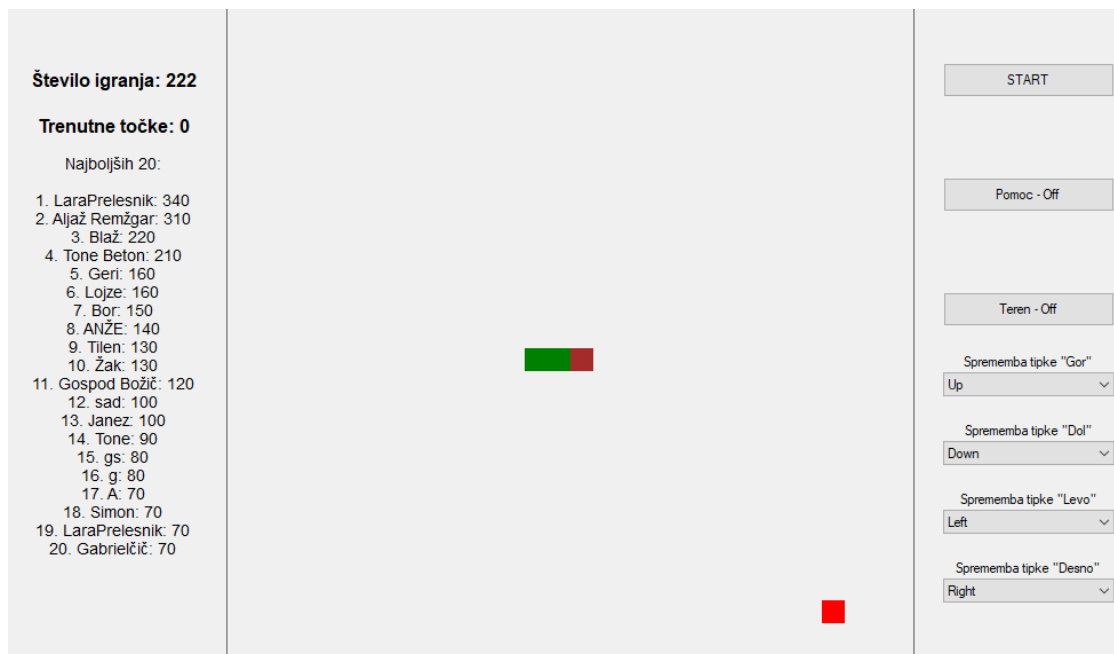


Vnesite uporabniško ime:

Prijava

Slika 1: Prijava

Ko smo enkrat notri zagledamo sledeče.



Število igranja: 222

Trenutne točke: 0

Najboljših 20:

1. LaraPrelesnik: 340
2. Aljaž Remžgar: 310
3. Blaž: 220
4. Tone Beton: 210
5. Geri: 160
6. Lojze: 160
7. Bor: 150
8. ANŽE: 140
9. Tilen: 130
10. Žak: 130
11. Gospod Božič: 120
12. sad: 100
13. Janez: 100
14. Tone: 90
15. gs: 80
16. g: 80
17. A: 70
18. Simon: 70
19. LaraPrelesnik: 70
20. Gabrielčič: 70

START

Pomoc - Off

Teren - Off

Sprememba tipke "Gor"
Up

Sprememba tipke "Dol"
Down

Sprememba tipke "Levo"
Left

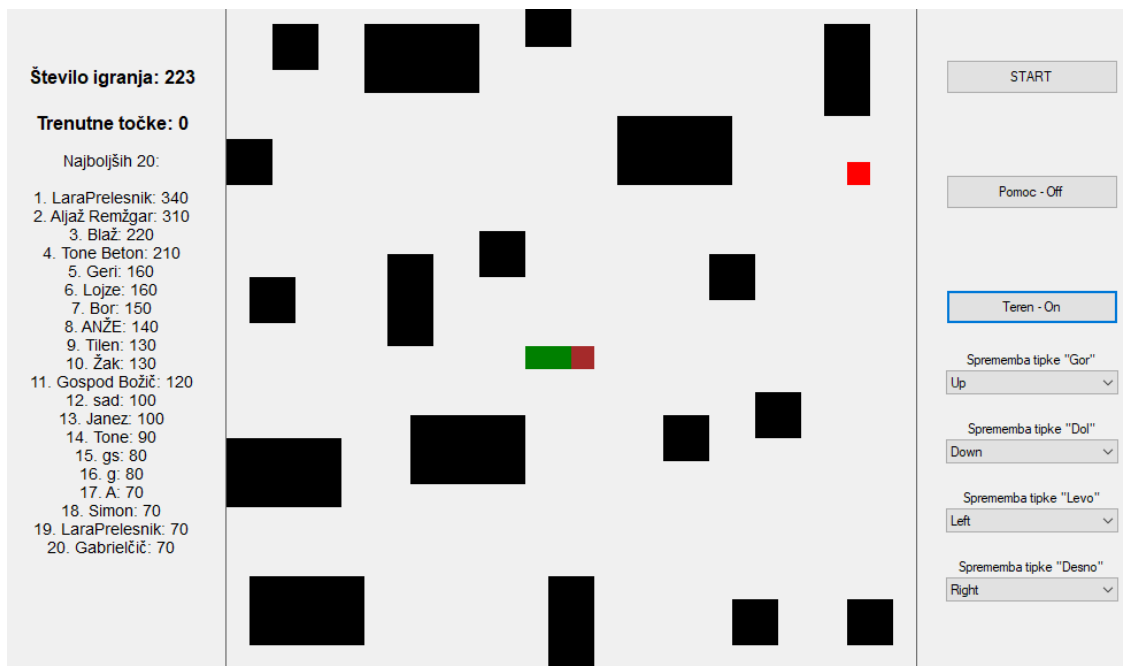
Sprememba tipke "Desno"
Right

Slika 2: Požrešna kačica

Na levi strani imamo ploščo in v zgornjem delu imamo števec, ki nam pove kolikokrat je bila že kačica igrana. Prav tako pa igra sledi, koliko točk smo v eni igri trenutno dosegli in ob koncu, pogleda, če so naše točke dovolj visoke, da se uvrstimo v lestvico, najboljših 20.

Kjer sem pa res povzpел igro na višjo raven, je ko sem dodal sledeče 3 nastavitve.

- **Prva nastavev** je pomoč. Ta nastavev omogoča igralcu, da lahko kačica se premika skozi zidove in se njeno življenje ne konča, vendar pride na drugi strani zida ven.
- **Druga nastavev** je teren. Ta nastavev ob kliku za igralca generira naključen teren, ki mu igro uteži. Tako da, če je osnovna igra prelahka, lahko vedno postane težja. Tukaj je primer terena:



Slika 3: Primer terena

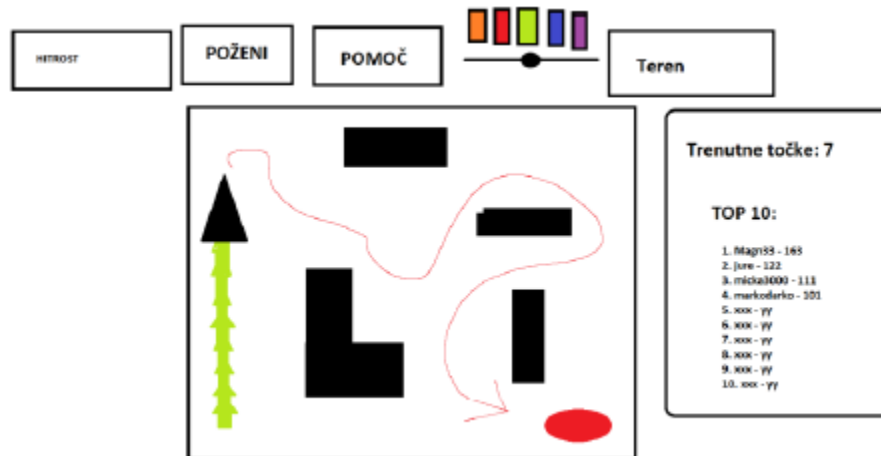
Če se kačica dotakne terena, se igra zaključi.

- **Tretja nastavev** je omogočanje nastavitve poljubnih kontrol za kačico. Lahko kliknemo na poljubno kontrolo na desni plošči in dobili bomo spisek vsek možnih kontrol, s katero lahko trenutno kontrolo zamenjamo za.

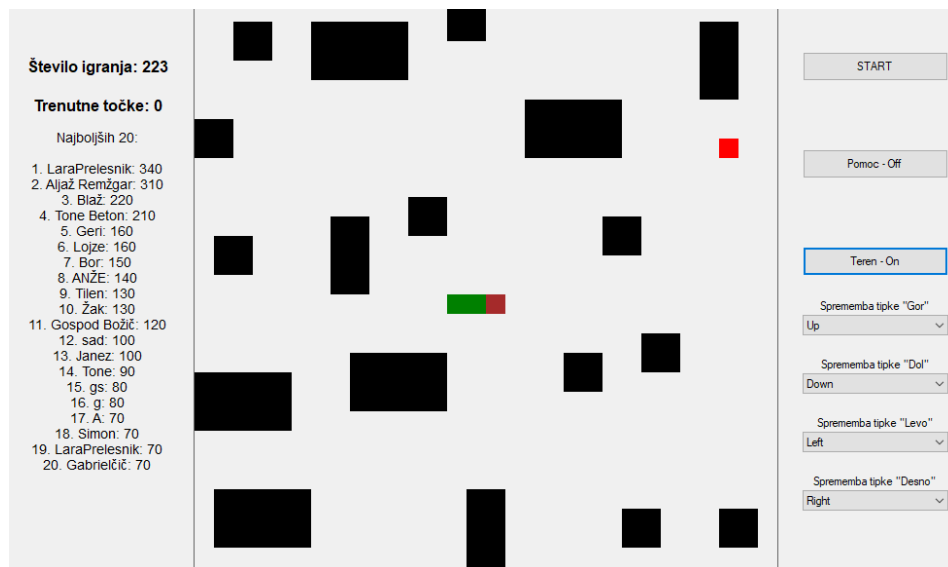
Ko imamo vse kontrole izbrane, kliknemo gumb start, ki bo omogočalo začetek premikanja kačice.

Razlika med izdelkom in prijavljenim opisom

Da bomo dobro razumeli razliko, je najbolje da si ogledam pred in po izgled izdelka.



Slika 4: Pred



Slika 5: Po

Kot vidimo je izdelek skoraj identičen kakor sem si ga predstavljal, vendar pa sem izpustil nastavitve hitrost, saj ob testiranju te nastavitve sem ugotovil, da je nastavev neuporabna, saj povečana hitrost kačice naredi igrico zelo težko za igrati še posebej skupaj z terenom.

Druga sprememba je pa nastavitev barve kačice. Zanj sem se kasneje odločil, da jo bom opustil, saj nisem videl smisla, da bi dodal to nastavitev, saj ne doda nič posebnega pri kodi ali pri igranju.

Glede na vse ostalo, pa je igrica taka, kot sem si jo zasnoval.

Opis ključnih prijemov in korakov

Prijava uporabnika

V tej metodi ustvarjamo prijavi obrazec, ki uporabnikom omogoča vnos uporabniškega imena pred začetkom igre. Najprej ustvarimo nov obrazec za prijavo. Uporabljamo `using` blok, da zagotovimo, da se obrazec pravilno sprost iz pomnilnika, ko se zapre. Obrazec nastavimo tako, da se odpre na sredini glavnega okna, določimo njegovo velikost in preprečimo, da bi uporabnik spreminjal njegovo velikost ali ga minimiziral.

Dodamo tudi gumb za prijavo, ki je postavljen pod besedilnim poljem. Ko uporabnik klikne na gumb, se sproži dogodek, ki preveri, ali je uporabniško ime veljavno (ni prazno ali vsebuje samo presledke). Če uporabniško ime ni veljavno, se prikaže sporočilo o napaki. Če je uporabniško ime veljavno, ga shranimo v spremenljivko `uporabniškoIme`, nastavimo rezultat obrazca na `DialogResult.OK` in obrazec se zapre.

Obrazec je prikazan kot modalno okno, kar pomeni, da uporabniki ne morejo komunicirati z glavno aplikacijo, dokler ne zaprejo prijavnega obrazca. To zagotavlja, da morajo uporabniki vnesti svoje uporabniško ime, preden lahko začnejo igrati igro.

Ta prijavi obrazec je potreben za shranjevanje doseženih točk v bazo podatkov pod uporabnikovim imenom, kar omogoča sledenje napredku in prikazovanje najboljših rezultatov.

```
private void Prijava()
{
    using (Form prijavniObrazec = new Form())
    {
        prijavniObrazec.StartPosition = FormStartPosition.CenterParent;
        prijavniObrazec.Size = new Size(300, 200);
        prijavniObrazec.FormBorderStyle = FormBorderStyle.FixedDialog;
        prijavniObrazec.MaximizeBox = false;
        prijavniObrazec.MinimizeBox = false;
        prijavniObrazec.ControlBox = false;

        Label prijavaNapis = new Label();
        prijavaNapis.Text = "Vnesite uporabniško ime:";
        prijavaNapis.Location = new Point(10, 20);
        prijavaNapis.Size = new Size(260, 30);
        prijavaNapis.TextAlign = ContentAlignment.MiddleCenter;
        prijavniObrazec.Controls.Add(prijavaNapis);

        TextBox prijavaTextBox = new TextBox();
        prijavaTextBox.Location = new Point(10, 60);
        prijavaTextBox.Size = new Size(260, 30);
        prijavniObrazec.Controls.Add(prijavaTextBox);

        Button prijavniGumb = new Button();
        prijavniGumb.Text = "Prijava";
        prijavniGumb.Location = new Point(100, 100);
        prijavniGumb.Click += (sender, e) => {
            if (string.IsNullOrWhiteSpace(prijavaTextBox.Text))
            {
                MessageBox.Show("Prosimo, vnesite veljavno uporabniško ime.", "Napaka", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            else
            {
                uporabniškoIme = prijavaTextBox.Text; // funkcija gumba
                prijavniObrazec.DialogResult = DialogResult.OK;
                prijavniObrazec.Close();
            }
        };
        prijavniObrazec.Controls.Add(prijavniGumb);

        prijavniObrazec.ShowDialog(); // ne moremo komunicirati z glavno aplikacijo, dokler ne zapremo tega okna.
    }
}
```

Slika 6: Prijava

Ustvari sadež

V tej metodi ustvarjamo nov sadež na igralnem polju. Metoda poskrbi, da se sadež ne pojavi na kači ali oviri, če je teren vklopljen. Za to imamo sestavljeni še drugi dve funkciji, ki skrbita za to. Igralno polje je definirano z velikostjo in položajem plošče za točke ter plošče za nastavitve, kar zagotavlja, da se sadež generira samo znotraj teh meja.

Celice so osnovne enote igre, ki določajo položaj in velikost kače ter sadeža na igralnem polju. Velikost celice je definirana z velikostCelice, kar pomeni, da je vsaka celica kvadrat s stranico določene dolžine (v tem primeru 20 pik). Celice omogočajo enostavno določanje položajev objektov na igralnem polju, saj se vsi objekti premikajo in generirajo v enotah celic.

Nato definiramo spremenljivko veljavnaPozicija, ki je na začetku nastavljena na false. Znotraj while zanke poskušamo ustvariti nov sadež na naključni lokaciji, dokler ne najdemo veljavne pozicije.

Znotraj zanke ustvarimo naključno točko sadez znotraj meja igralnega polja. Točko pretvorimo v pravokotnik kvadratSadeza, ki ima velikost ene celice.

Preverimo, ali je pozicija sadeža veljavna. To storimo tako, da preverimo, ali pravokotnik kvadratSadeza leži znotraj igralnega polja, da se sadež ne prekriva s kačo (!Sadez_v_kaci()) in da se ne prekriva z ovirami, če je teren vklopljen (!SadezNaOviro(kvadratSadeza)). Če so vsi ti pogoji izpolnjeni, nastavimo veljavnaPozicija na true, kar ustavi zanko.

```
private void UstvariSadez()
{
    Random rnd = new Random();
    Rectangle kvadratSadeza;
    Rectangle igralnoPolje = new Rectangle(ploscaZaTocke.Width, 0, this.ClientSize.Width - ploscaZaTocke.Width - ploscaZaNastavitve.Width, this.ClientSize.Height);

    bool veljavnaPozicija = false;

    while (!veljavnaPozicija)
    {
        // poskusimo nov sadez ustvariti
        sadez = new Point(rnd.Next(ploscaZaTocke.Width / velikostCelice, (this.ClientSize.Width - ploscaZaNastavitve.Width) / velikostCelice), rnd.Next(0, this.ClientSize.Height / velikostCelice));

        kvadratSadeza = new Rectangle(sadez.X * velikostCelice, sadez.Y * velikostCelice, velikostCelice, velikostCelice);

        // preverimo, da je sadez na veljavni poziciji
        veljavnaPozicija = igralnoPolje.Contains(kvadratSadeza) && !Sadez_v_kaci() && (!terenVklopljen || !SadezNaOviro(kvadratSadeza));
    }
}
```

Slika 7: Ustvari sadež

Ustvarjanje ovir

V tej metodi ustvarjamo ovire na igralnem polju. Metoda poskrbi, da se ovire generirajo na veljavnih pozicijah in se ne prekrivajo s kačo, sadežem ali drugimi ovirami.

Definiramo velikosti ovir in količine posameznih tipov ovir. V tabeli velikosti določimo širino in višino posameznih tipov ovir, medtem ko tabela kolicinaOvire določa, koliko ovir vsakega tipa želimo generirati. Znotraj prve for zanke iteriramo skozi različne tipe ovir. Za vsak tip ovire določimo širino in višino iz tabele velikosti. Znotraj druge for zanke iteriramo skozi število ovir za

trenutni tip. Za vsako oviro naredimo pravokotnik `ovira` in nastavimo spremenljivko `oviraJeOvirana` na `true`.

Znotraj `while` zanke poskušamo ustvariti oviro na naključni lokaciji, dokler ne najdemo veljavne pozicije. Ustvarimo naključno točko `Lokacija` znotraj meja igralnega polja. Točko pretvorimo v pravokotnik `ovira`, ki ima določeno širino in višino.

Preverimo, ali je pozicija ovire veljavna. To storimo tako, da preverimo, ali pravokotnik `ovira` ne prekriva drugih ovir, kaže ali sadeža (`oviraJeOvirana = Oviran(ovira)`). Če so vsi ti pogoji izpolnjeni, nastavimo `oviraJeOvirana` na `false`, kar ustavi zanko. Na koncu dodamo veljavno oviro v seznam `ovire`.

```
private void UstvariOvire()
{
    Random rnd = new Random();
    Rectangle igralnoPolje = new Rectangle(ploscaZaTocke.Width, 0, this.ClientSize.Width - ploscaZaTocke.Width - ploscaZaNastavitve.Width, this.ClientSize.Height);

    int[] velikosti = { 2, 2, 5, 3, 2, 4 }; // velikosti ovir, torej prva je 2x2, druga 5x3 in tretji tip ovire je 2x4
    int[] kolicinaOvire = { 10, 5, 3 }; // število ovir različnih tipov

    for (int i = 0; i < kolicinaOvire.Length; i++) // lahko razumemo tako, da i predstavlja posamezno oviro, npr i = 0 predstavlja 10 ovir oblike 2x2
    {
        int širina = velikosti[2 * i];
        int višina = velikosti[2 * i + 1];

        for (int j = 0; j < kolicinaOvire[i]; j++)
        {
            Rectangle ovira = new Rectangle();
            bool oviraJeOvirana = true;

            while (oviraJeOvirana)
            {
                Point Lokacija = new Point(rnd.Next(ploscaZaTocke.Width / velikostCelice, (this.ClientSize.Width - ploscaZaNastavitve.Width - širina * velikostCelice) / velikostCelice), rnd.Next(0, (this.ClientSize.Height - višina * velikostCelice) / velikostCelice));
                ovira = new Rectangle(Lokacija.X * velikostCelice, Lokacija.Y * velikostCelice, širina * velikostCelice, višina * velikostCelice);
                oviraJeOvirana = Oviran(ovira); // igralnoPolje.Contains(ovira) ||
            }

            ovire.Add(ovira);
        }
    }
}
```

Slika 8: Ustvari ovire

Preveri trk

V tej metodi preverjamo, ali kača trči v robove okna, svoje telo, plošče ali ovire. Najprej preverimo, če je glava kače zadela rob okna, in če je temu tako, zaključimo igro. Nato preverimo, ali se glava kače prekriva s katerimkoli delom njenega telesa, kar prav tako pomeni konec igre.

Nadalje preverimo trk s ploščami za točke in nastavitve. Če ugotovimo, da se glava kače dotika katere koli plošče, igro zaključimo. Če je teren vklopljen, preverimo še, ali kača trči v katero od ovir na igralnem polju. Če je katerakoli od teh preveritev pozitivna, igro zaključimo.

Metoda za preverjanje trkov s ploščami ugotavlja, ali se glava kače dotika plošče za točke ali nastavitve. Najprej preverimo, ali je pomoč vklopljena. Če je, dovolimo prehod kače skozi levi ali desni rob igralnega polja. Če pomoč ni vklopljena, preverimo, ali se glava kače dotika katere koli plošče. Če ugotovimo, da se dotika, to pomeni trk in igro zaključimo.

```

// reference
private void PreveriTrk()
{
    Point glava = kaca[0];

    // preverimo trk z robom
    if (glava.X < 0 || glava.Y < 0 || glava.X >= this.ClientSize.Width / velikostCelice || glava.Y >= this.ClientSize.Height / velikostCelice)
    {
        KonecIgre();
        return; // imamo return, da ko je konec igre da metoda ne preverja naprej za druge opcije
    }

    // preverimo trk same s sabo
    for (int i = 1; i < kaca.Count; i++)
    {
        if (glava == kaca[i])
        {
            KonecIgre();
            return;
        }
    }

    // preverimo trk s ploščami
    if (PreveriTrkPlosc(glava))
    {
        KonecIgre();
        return;
    }

    // preverimo trk z ovirami, če je teren vklopljen
    if (terenVklopljen)
    {
        Rectangle kvadratGlave = new Rectangle(glava.X * velikostCelice, glava.Y * velikostCelice, velikostCelice, velikostCelice);
        foreach (var ovira in ovire)
        {
            if (kvadratGlave.Intersects(ovira))
            {
                KonecIgre();
                return;
            }
        }
    }
}

```

Slika 9: Preveri trk

Premikanje

V tej metodi upravljamo premikanje kače na igralnem polju. Premikanje kače se simulira tako, da se nova pozicija glave doda na začetek seznama, zadnji del kače pa se odstrani, razen če kača poje sadež.

Najprej pridobimo trenutni položaj glave kače in ga shranimo v spremenljivko `glava`. Nato ustvarimo kopijo tega položaja v spremenljivki `novaGlava`, ki jo bomo posodobili glede na smer gibanja.

Preverimo, v katero smer se kača trenutno giba (gor, dol, levo, desno), in ustrezno posodobimo koordinati `x` ali `y` za glavo kače.

Če se kača giba gor, zmanjšamo `y` koordinato za 1. Če glava kače preseže zgornji rob igralnega polja, preverimo, ali je pomoč vklopljena. Če je, premaknemo glavo kače na spodnji rob igralnega polja. Če pomoč ni vklopljena, končamo igro. Na tak način naredimo tudi za ostale možnosti.

Nato dodamo novo pozicijo glave kače na začetek seznama `kaca`, kar simulira premikanje naprej. Če nova glava kače sovpada s pozicijo sadeža, povečamo trenutne točke za 10, posodobimo prikaz točk in ustvarimo nov sadež. Ker v tem primeru ne odstranimo zadnjega dela kače, kača dejansko raste. Če nova glava kače ni na poziciji sadeža, odstranimo zadnji del kače s seznamom `kaca`, kar ohranja dolžino kače nespremenjeno in ustrezno simulira premikanje kače po igralnem polju.

```

private void Premikanje()
{
    Point glava = kaca[0];
    Point novaGlava = glava;

    if (smer == "up")
    {
        novaGlava.Y -= 1;
        if (novaGlava.Y < 0) // prehod skozi zgornji rob
        {
            if (pomoc.Text == "Pomoc - On")
            {
                novaGlava.Y = (this.ClientSize.Height / velikostCelice) - 1; // premaknemo kacic na spodni rob in se od tam premika naprej
            }
            else
            {
                KonecIgre(); // igra se konča, če ni pomoči
                return;
            }
        }
    }
    else if (smer == "down")
    {
        novaGlava.Y += 1;
        if (novaGlava.Y >= this.ClientSize.Height / velikostCelice) // prehod skozi spodnji rob
        {
            if (pomoc.Text == "Pomoc - On")
            {
                novaGlava.Y = 0;
            }
            else
            {
                KonecIgre();
                return;
            }
        }
    }
    else if (smer == "left")
    {
        novaGlava.X -= 1;
        if (novaGlava.X < ploscaZaTocke.Width / velikostCelice) // prehod skozi levi rob
        {
            if (pomoc.Text == "Pomoc - On")
            {
                novaGlava.X = (this.ClientSize.Width - ploscaZaNastavitve.Width) / velikostCelice - 1;
            }
        }
    }
}

```

Slika 10: Premikanje

```

    }
    else if (smer == "right")
    {
        novaGlava.X += 1;
        if (novaGlava.X >= (this.ClientSize.Width - ploscaZaNastavitve.Width) / velikostCelice) // prehod skozi desni rob
        {
            if (pomoc.Text == "Pomoc - On")
            {
                novaGlava.X = ploscaZaTocke.Width / velikostCelice;
            }
            else
            {
                KonecIgre();
                return;
            }
        }
    }
}

// simulacija premikanje kace
kaca.Insert(0, novaGlava);

if (novaGlava == sadez)
{
    trenutneTocke += 10;
    trenutneTockeLabel.Text = "Trenutne točke: " + trenutneTocke; //ker tukaj ne odstranimo zadnjega dela kace insamo dodamo kaco, tukaj simuliramo povecanje kace
    UstvariSadez();
}
else
{
    kaca.RemoveAt(kaca.Count - 1);
}
}
}

```

Slika 11: Premikanje