Dokumentacija igre Vješala

Ana Hodžić, Iva Vuletić, Luka Žigolić, Mario Borna Mjertan

rujan 2025.

1 Uvod

U ovom dokumentu opisan je kod napravljen za igru Vješala kao završni projekt kolegija Multimedijski sustavi.

Kod u IgreVjesala.pde je podijeljen na 5 dijelova kako bi olakšali čitatelju snalaženje, a to su globalne varijable, osnovne funkcije, pomoćne funkcije, mrežne funkcije i funkcije obrada unosa. U ovoj dokumentaciji opis projekta je podijeljen kronološki po dodanim funkcionalnostima. Prvo smo postavili osnovnu verziju igre s njenim izbornikom i glavnom logikom igre za singleplayer opciju, a zatim dodali funkcionalnosti crtanja vješala kao vizualni prikaz gubljenja pokušaja u igri te implementirali multiplayer. Finalno, igri smo dodali opciju promjene postavki.

2 Funkcionalnosti igre

2.1 Singleplayer verzija igre

Igranje je podijeljeno u nekoliko glavnih stanja između kojih se igrač kreće: glavni izbornik, izbornik levela, sama igra, te ekran rezultata. Na taj način igra ima strukturu sličnu ostalim računalnim igrama - igrač započinje u meniju, odabire razinu, igra, dobije rezultat te na kraju odlučuje hoće li nastaviti dalje.

U glavnom meniju se prikazuju osnovne opcije "Singleplayer", "Multiplayer", "Postavke" i "Izlaz". Klikom na opciju "Singleplayer", igrač dolazi u izbornik levela gdje bira težinu. Nakon što odabere level, igra prelazi u glavni dio - pogađanje skrivene riječi. Riječ je zadana unaprijed, a igrač pokušava odgonetnuti riječ pogađanjem jednog po jednog slova. Kod svake pogreške se postupno iscrtava figura čovjeka. Igra završava kada igrač ili pogodi cijelu riječ ili napravi previše pogrešaka.

Nakon završetka igre, pojavljuje se ekran sa ispisanom porukom o pobjedi ili gubitku, zelene ili crvene boje, ovisno o rezultatu. Ispod poruke se nalazi opcija "Natrag na level menu" na koju igrač može kliknuti kada je spreman nastaviti.

Za iscrtavanje izbornika koristili smo funkciju draw() koja ovisno o vrijdenosti True ili False varijable (koja određuje opciju) poziva pomoćne funkcije za prikaz opcija iz izbornika, rezultat,...

```
void draw() {
  background(200, 220, 255);
  hoverIndex = -1;

  netUpdate();

  if (prikaziRezultat) {
    prikaziRezultatEkran();
  } else if (igraAktivna) {
    prikaziIgru();
  } else if (netEnterWord) {
  prikaziNetUnosRijeci();
  } else if (multiplayerMenu) {
    prikaziOpcije(multiplayerOpcije);
  } else if (levelMenu) {
    prikaziOpcije(levelOpcije);
  } else if (singleplayerMenu) {
    prikaziOpcije(singleplayerOpcije);
  } else {
    prikaziOpcije(opcije);
  }
}
```

Funkcija prikaziOpcije(String[] izbornik) služi za upravljanje izbornikom opcija (npr. glavnog menija ili level menija). Za svaku stavku menija izračunava njezinu poziciju i ako se pokazivač miša nalazi unutar okvira teksta neke opcije, sprema indeks te opcije u varijablu hoverIndex. To znači da zna "koja opcija je trenutno pod mišem".

```
void prikaziOpcije(String[] izbornik) {
  for (int i = 0; i < izbornik.length; i++) {
    float x = width / 2;
    float y = 100 + i * 50;
    float w = textWidth(izbornik[i]);
    float h = 32;

    boolean isHovered = mouseX > x - w / 2 && mouseX < x + w / 2 &&
                        mouseY > y - h / 2 && mouseY < y + h / 2;
    if (isHovered) hoverIndex = i;
  }

  if (hoverIndex != -1) {
    mainMenu = hoverIndex;
  }

  for (int i = 0; i < izbornik.length; i++) {
    float x = width / 2;
    float y = 100 + i * 50;
```

```
    if (i == mainMenu) {
      fill(255, 100, 100);
    } else {
      fill(0);
    }
    text(izbornik[i], x, y);
  }
}
```

Funkcija inicijalizirajRijec() postavlja novu riječ ovisno o odabranom levelu. Odabire slučajnu riječ iz unaprijed definirane liste, te pripremi niz znakova s podvlakama umjesto slova, izbriše prethodno unesena slova i resetira broj pokušaja na početnih 6.

```
void inicijalizirajRijec() {
  String[] trenutnaLista;
  if (odabraniLevel == 1) {
    trenutnaLista = rijeciLevel1;
  } else if (odabraniLevel == 2) {
    trenutnaLista = rijeciLevel2;
  } else {
    trenutnaLista = rijeciLevel3;
  }
  rijec = trenutnaLista[(int) random(trenutnaLista.length)];
  prikazano = new char[rijec.length()];
  for (int i = 0; i < prikazano.length; i++) {
    prikazano[i] = '_';
  }
  unesenaSlova.clear();
  pokusaji = 6;
}
```

Funkcija provjeriUnos(char unos) provjerava je li igrač pogodio slovo. Prolazi kroz cijelu skrivenu riječ i ako uneseno slovo postoji, otkriva ga na odgovarajućem mjestu. Ako se slovo ne nalazi u riječi, a igrač još ima pokušaja, smanjuje se broj preostalih pokušaja.

```
void provjeriUnos(char unos) {
  boolean pogodak = false;
  for (int i = 0; i < rijec.length(); i++) {
    if (rijec.charAt(i) == unos) {
      prikazano[i] = unos;
      pogodak = true;
    }
  }
  if (!pogodak && pokusaji > 0) {
    pokusaji--;
```

```
  }
}
```

Funkcija prikaziIgru() zadužena je za prikaz cijelog stanja igre dok igrač igra. Prvo crta vješala, zatim prikazuje dosad pogođena slova i crtice za neotkrivena slova, a ispod popis svih unesenih slova. Ako je igra u multiplayer modu, ispisuje trenutni rezultat i uloge igrača (tko postavlja riječ, a tko pogađa). Ako je igra singleplayer, prikazuje koji je level odabran. Na kraju se provjeravaju uvjeti završetka igre: ako su pokušaji potrošeni, igra završava porazom, a ako su slova pogođena, onda pobjedom.

```
void prikaziIgru() {
  fill(0);

  //crtanje vješala
  int promasaji = 6 - pokusaji;
  crtajVjesala(promasaji);

  //pogođena slova i _
  String prikazTekst = "";
  boolean svePogodeno = true;
  for (char c : prikazano) {
    prikazTekst += c + " ";
    if (c == '_') svePogodeno = false;
  }

  float margin = 40;
  pushStyle();
  textAlign(RIGHT, TOP);
  text(prikazTekst, width - margin, 150);
  popStyle();

  pushStyle();
  textAlign(RIGHT, TOP);
  float yLabel = 230;
  float boxW = min(420, width - 2*margin);
  float xBox = width - margin - boxW;
  fill(0);
  text("Unesena slova:", width - margin, yLabel);
  float lh = textAscent() + textDescent() + 6;
  textLeading(lh);
  String slovaStr = unesenaSlovaTekst();
  text(slovaStr, xBox, yLabel + lh, boxW, height - (yLabel + lh) - margin);
  popStyle();

  if(netHost || netJoin){
    //multiplayer
    pushStyle();
    textAlign(RIGHT, TOP);
```

```
    textSize(20);
    String uloge = "Postavlja: " + (setterIsHost ? "HOST" : "CLIENT") +
        " | Pogađa: " + (setterIsHost ? "CLIENT" : "HOST");
    text("Bodovi HOST " + scoreHost + " - " + scoreClient + " CLIENT | "
        + uloge, width - margin, 20);
    text(uloge, width - margin, 50);
    popStyle();
  } else {
    //singleplayer
    pushStyle();
    textAlign(RIGHT, TOP);
    text("Level: " + odabraniLevel, width - margin, 320);
    popStyle();
  }

  if(!(netHost || netJoin)) {
    if (pokusaji <= 0) {
      prikaziRezultat = true;
      pobjeda = false;
      igraAktivna = false;
      mainMenu = -1;
    } else if (svePogodeno) {
      prikaziRezultat = true;
      pobjeda = true;
      igraAktivna = false;
      mainMenu = -1;
    }
  }

}
```

Funkcija prikaziRezultatEkran() služi za prikaz završnog ekrana igre na način opisan na početku odjeljka.

```
void prikaziRezultatEkran() {
  fill(pobjeda ? color(0, 150, 0) : color(255, 0, 0));
  text(pobjeda ? "Pobijedio si!" : "Izgubio si!", width / 2, 60);

  String[] rezultatOpcije = (netHost || netJoin) ?
    new String[]{"Natrag na multiplayer menu"} :
    new String[]{"Natrag na level menu"};
  prikaziOpcije(rezultatOpcije);
}
```

2.2 Crtanje vješala

Vješalo je nacrtano na lijevom dijelu ekrana i sa svakom greškom igrača se nacrta novi dio tijela. Igrač ima pravo 6 puta pogriješiti te kad se nacrtaju glava, trup i udovi, igrač gubi igru.

Postojećem kodu koji ima brojač za pokušaje smo dodali varijablu promašaji koja je brojač povezan sa crtanjem vješala. Ovisno koliko je pokušaja potrošeno, ovisi koliko udova crtamo.

Glavna funkcija koja nam pomaže u crtanju vješala je crtajVjesala(promasaji).

```
void crtajVjesala(int promasaji) {
  int x0 = 80;
  int y0 = 340;
  int visina = 250;
  int ruka = 80;
  int uze = 40;

  stroke(0);
  strokeWeight(4);

  line(x0, y0, x0 + 100, y0);
  line(x0 + 20, y0, x0 + 20, y0 - visina);
  line(x0 + 20, y0 - visina, x0 + 20 + ruka, y0 - visina);
  line(x0 + 20 + ruka, y0 - visina, x0 + 20 + ruka, y0 - visina + uze);

  int cx = x0 + 20 + ruka;
  int cy = y0 - visina + uze + 20;

  if (promasaji >= 1) {
    noFill();
    ellipse(cx, cy, 40, 40);
  }
  if (promasaji >= 2) {
    line(cx, cy + 20, cx, cy + 20 + 60);
  }
  if (promasaji >= 3) {
    line(cx, cy + 30, cx - 35, cy + 55);
  }
  if (promasaji >= 4) {
    line(cx, cy + 30, cx + 35, cy + 55);
  }
  if (promasaji >= 5) {
    line(cx, cy + 80, cx - 30, cy + 120);
  }
  if (promasaji >= 6) {
    line(cx, cy + 80, cx + 30, cy + 120);
  }
}
```

Tu funkciju pozivamo u prikaziIgru().

2.3 Multiplayer

Igrač u multiplayer modu može odabrati želi li biti onaj koji postavlja riječ ili onaj koji ju pogađa. Ako onaj igrač koji pogađa riječ uspije pogoditi, onda dobiva bod i nastavlja biti onaj koji pogađa sve dok ne pogriješi. Kad igrač pogriješi, onaj koji je zadao riječ dobiva bod i onda je njegov red da pogađa. Igra se dok netko ne skupi 5 bodova. Multiplayer opcija je napravljena tako da se od ove verzije koda, IgraVjesala.pde, napravi kopija i pokrenu se obje u Processing-u. Dakle, igra je složena da se lokalno igra na jednom računalu. U jednom prozoru igrač upiše riječ, a u drugom pogađa. Na oba prozora je vidljivo stanje igre i trenutni bodovi oba igrača.

Za multiplayer opciju morali smo uključiti i import processing.net.*; koji nam omogućava mrežne igranje. Također, postavili smo port 50007 i lokalnu ip adresu 127.0.0.1, a za logiku su nam potrebne i zastavice.

```
Server srv;
Client cli;
final int PORT = 50007;
final String LOCAL_IP= "127.0.0.1";

boolean netHost = false;
boolean netJoin = false;
boolean netConnected = false;
boolean netIpInput = false;
boolean netEnterWord = false;

String wordBuffer = ""; //unos riječi od host-a

boolean setterIsHost = true; //host je uvijek postavljač riječi na početku

int scoreHost = 0;
int scoreClient = 0;
final int WIN_SCORE = 5; //igra se do 5
```

Multiplayer opcija igre započinje s time da igrač bira, na izbornom meni-ju, hoće li postavljati riječ ili pogađati.

```
String[] multiplayerOpcije = {"Postavi riječ", "Pogađaj", "Natrag"};
```

Ukoliko igrač odabere postavljanje, poziva se startHost koji pokreće server i postavlja zastavice na stranu host-a, a ako odabere pogađanje onda se poziva startJoinLocal() i time se spaja na server te postavlja zastavice na stranu klijenta.

```
void startHost() {
  closeNetwork();
  try {
    srv = new Server(this, PORT);
```

```
    println("Server pokrenut na portu " + PORT);
  } catch (Exception e) {
    println("Ne mogu pokrenuti server: " + e.getMessage());
  }
  netHost = true;
  netJoin = false;
  netConnected = false;

  scoreHost = scoreClient = 0; //postavlja rezultate na početak
  setterIsHost = true;

  wordBuffer = ""; //isprazni riječ da bi mogli upisati novu
  netEnterWord = true;   // host odmah postavlja prvu riječ
}

void startJoinLocal() {
  closeNetwork();
  netJoin = true;
  netHost = false;
  netConnected = false;

  try {
    cli = new Client(this, LOCAL_IP, PORT);
    netConnected = true;
    println("JOIN (lokalno): spojeno na " + LOCAL_IP + ":" + PORT);
    cli.write("HELLO\n");
  } catch (Exception e) {
    println("JOIN (lokalno) neuspjelo: " + e.getMessage());
    netConnected = false;
  }

  multiplayerMenu = false;
  mainMenu = -1;
  igraAktivna = false;
}
```

Oblici **poruka** koji se pojavljuju u kodu su sljedeći:

(1) klijent ->server

GUESS|SLOVO šalje klijent serveru kad je setter host

SET|RIJEČ šalje klijent kad je on setter

(2) server ->klijent

STATE|<MASK>|<POKUSAJI>|<SLOVA>|HOST-A>|<SCORE KLIJENTA>|<SETTER IS HOST >

Glavna mrežna petlja se nalazi u netUpdate() koja se poziva sa svakim pozivom funkcije draw().

Ta funkcija ima sljedeće radnje:

(1) Host prihvaća spajanje

Host ili prihvaća spajanje klijenta ili ako je igra u tijeku, vraća stanje igre pozivom netSendState()

(2) Host čita klijentovu poruku

Pozivom handleClientMsg() se obrađuje poruka koju je klijent poslao

(3) Klijent čita poruku od host-a

Pozivom handleServerMsg() se obrađuje poruka koju je host poslao

(4) Prekid veze

Host **postavlja** riječ na ekranu koji je zadan funkcijom prikaziNetUnosRijeci() i time se riječ sprema u wordBuffer. Zatim se u funkciji keyPressed() provjerava zastavica za unesenu riječ te se poziva serverNewRoundSetWord(). Kada klijent postavlja riječ, poziva se handleClientMsg() u kojoj se također pozove serverNewRoundSetWord().

```
void serverNewRoundSetWord(String zadana) {
  inicijalizirajRijecZadana(zadana);
  igraAktivna = true;
  netEnterWord = false;
  netSendState();
}
```

Pogađa onaj koji nije setter (guesserIsHost zastavica), a u keyPressed() funkciji se poziva serverGuess() koja provjerava unos i ukoliko su potrošeni svi pokušaji ili sva slova pogođena, poziva funkciju za kraj serverEndRound() ili netSendState() za slanje poruke koja sadržava sve info.

```
//pogađanje
void serverGuess(char unos) {
  if (unesenaSlova.contains(unos)) return;
  unesenaSlova.add(unos);
  provjeriUnos(unos);

  boolean svePogodeno = true;
  for (char c : prikazano) if (c == '_') {svePogodeno = false; break;}

  if (pokusaji <= 0 || svePogodeno) {
    boolean guesserWon = svePogodeno;
    serverEndRound(guesserWon);
  } else {
    netSendState();
  }
}

//završetak runde + bodovi + odluka o sljedećem setteru
```

```
void serverEndRound(boolean guesserWon) {
  boolean guesserIsHost = !setterIsHost;

  if (guesserWon) {
    if (guesserIsHost) scoreHost++;
    else scoreClient++;
    // setter ostaje isti
  } else {
    // pogađač izgubio -> bod postavljaču, uloge se zamijene
    if (setterIsHost) scoreHost++;
    else scoreClient++;
    setterIsHost = !setterIsHost; // zamjena uloga postavljača
  }

  boolean matchOver = (scoreHost >= WIN_SCORE) ||
  (scoreClient >= WIN_SCORE);

  // pošalji rezultat klijentu
  if (cli != null) {
    cli.write("RESULT|" + (guesserWon ? 1 : 0) + "|" + rijec + "|" +
              scoreHost + "|" + scoreClient + "|" +
              (setterIsHost ? 1 : 0) + "|" +
              (matchOver ? 1 : 0) + "\n");
  }

  if (matchOver) {
    // prikaži rezultat lokalno
    igraAktivna = false;
    prikaziRezultat = true;
    pobjeda = (scoreHost >= WIN_SCORE);
    return;
  }

  // priprema za sljedeću rundu:tko postavlja neka upiše novu riječ
  igraAktivna = false;
  unesenaSlova.clear();
  pokusaji = 6;

  if (setterIsHost) {
    // host postavlja -> lokalni ekran za unos
    wordBuffer = "";
    netEnterWord = true;
  } else {
    // klijent postavlja
  }
}
```

Osim toga, važno je još napomenuti da u funkciji keyPressed() ovisno o zastavicama netEnterWord i igraAktivna se događaju radnje unosa tajne riječi

te pogađanje tj. radnje koje smo prethodno objasnili.

2.4 Promjena postavki

Unutar postavki igrač može kontrolirati nekoliko aspekata. Može mijenjati između svijetlog i tamnog načina rada, paliti i gasiti glazbu te mijenjati font slova korištenih u igri. U nastavku je prikaz globalnih varijabli korištenih za rad unutar postavki. Koristimo jednostavne varijable kako bi pratili promjenu u postavkama. Također je važno napomenuti da koristimo biblioteku sound, a navedeni fontovi bi trebali biti standardno dostupni unutar Processing-a.

```
boolean darkTheme = false;
PFont currentFont;
boolean musicOn = false;
SoundFile bgMusic;  // iz biblioteke "Sound"

String[] settingsOpcije = {"Tema: ", "Font: ", "Glazba: ", "Natrag"};
int currentFontIndex = 0;
String[] availableFonts = {"Arial", "Cambria", "Georgia"};
```

U početku se nalazimo u svijetloj temi, glazba je isključena i početni font nam je Arial. Unutar postavki zadržavamo funkcionalnosti koje smo imali i unutar glavnog menija (kao što je npr. promjena boje pri hover-anju miša iznad postavke)

```
void setup() {
  size(600, 400);
  textAlign(CENTER, CENTER);
  textSize(32);

  currentFont = createFont(availableFonts[currentFontIndex], 32);
  textFont(currentFont);

  bgMusic = new SoundFile(this, "music.mp3");
  bgMusic.loop();
  bgMusic.amp(0);
}

void prikaziPostavke() {
  hoverIndex = -1; // reset

  for (int i = 0; i < settingsOpcije.length; i++) {
    float x = width / 2;
    float y = 100 + i * 50;
    float w = textWidth(settingsOpcije[i] + "XXX"); // rezerva za dulji tekst
    float h = 32;

    // provjera miša
```

```
    boolean isHovered = mouseX > x - w / 2 && mouseX < x + w / 2 &&
                        mouseY > y - h / 2 && mouseY < y + h / 2;
    if (isHovered) hoverIndex = i;
  }

  if (hoverIndex != -1) {
    mainMenu = hoverIndex;
  }

  for (int i = 0; i < settingsOpcije.length; i++) {
    float x = width / 2;
    float y = 100 + i * 50;

    String textOpcija = settingsOpcije[i];
    if (i == 0) textOpcija += darkTheme ? "Tamna" : "Svijetla";
    if (i == 1) textOpcija += availableFonts[currentFontIndex];
    if (i == 2) textOpcija += musicOn ? "ON" : "OFF";

    if (i == mainMenu) fill(255, 100, 100);
    else fill(darkTheme ? 230 : 0);

    text(textOpcija, x, y);
  }
}
```

Unutar mousePressed() funkcije imamo glavni dio postavki, ovdje reagiramo na sve moguće klikove i prema njima radimo potrebne promjene unutar postavki. U slučaju promjene fonta pomičemo se za jedno mjesto u desno unutar naše liste fontova, pazeći pri tome da ostanemo unutar postojećih indexa. Glazbu jednostavno palimo ili gasimo korištenjem funkcije za glasnoću (ovisno o prijašnjem stanju), a promjena načina rada u tamni ili svijetli način je implementirana kroz sve draw() funkcije i funkcije crtanja/prikaza. Stoga nam je ovdje bitno samo negirati našu globalnu varijablu darkTheme kako bi se prebacili u novi način rada.

```
else if (settingsMenu) {
  if (mainMenu == 0) {
    darkTheme = !darkTheme;
  } else if (mainMenu == 1) {
    currentFontIndex = (currentFontIndex + 1) % availableFonts.length;
    currentFont = createFont(availableFonts[currentFontIndex], 32);
    textFont(currentFont);
  } else if (mainMenu == 2) {
    musicOn = !musicOn;
    if (musicOn) bgMusic.amp(0.5);
    else bgMusic.amp(0);
  } else if (mainMenu == 3) {
    settingsMenu = false;
    mainMenu = 0;
```

```
    }
  }
```