Opis projekta

Bubble Trouble

Dominik Mikulčić, Lorena Novak, Petra Škrabo, Tihana Štifanić, Katarina Šupe

Retro igra izrađena u Processingu napravljena po uzoru na igru hrvatskog kreatora Krešimira Cvitanovića. Igrač je predstavljen likom koji puca u mjehuriće koji odskaču od tlo i postaju manji ukoliko ih igrač pogodi. Svaki put kada igrač pogodi mjehurić, on se razdvoji u nekoliko manjih, a na igraču je da pripazi da ga ti manji mjehurići također ne pogode. Svaki level ima vremenski limit u kojem igrač mora pokušati uništiti sve mjehuriće što brže.

Predviđeni datum završetka rada: 1.4.2021.

Poveznica na projekt na Githubu: <https://github.com/katarinasupe/Bubble-Trouble>

Dnevnik rada

* 7. ožujka 2021. (Tihana Štifanić):
* Stvaranje klase *Player* koja predstavlja igrača. Igrač je određen svojom pozicijom *position* (po *x*-koordinati, kako je pomicanje moguće samo u tom smjeru) i brojem preostalih života *lives*. U klasi se također pamte podaci o igračevom koplju (trenutna pozicija glave koplja određena s *xSpear* i *ySpear* te neke pomoćne varijable koje označavaju je li koplje trenutno aktivno i u kojem smjeru se trenutno pomiče). Koplje igrača se aktivira spacebarom, a igrač njime može „razbijati“ loptice koje su trenutno na ekranu. Također su implementirane metode za iscrtavanje igrača i koplja *draw()*, metoda za pomicanje igrača za *x* piksela *move()* te metoda za ažuriranje *update()* u kojoj se trenutno samo ažurira pozicija glave koplja (kada je koplje aktivno, glava se prvo pomiče prema gore dok ne udari u gornji rub prozora, a tada prema dolje dok ne udari u donji rub, kad postaje neaktivno). Implementirane su i metode za resetiranje pozicije igrača *resetPosition()* te resetiranje koplja *resetSpear()*. Trenutno su sve metode implementirane sa samo jednim igračem u vidu, pa ih treba prilagoditi za igru s dva igrača. – 3 sata
* 8. ožujka 2021. (Tihana Štifanić):
  + Stvaranje klase *Ball* koja predstavlja lopticu. Loptica je određena svojim središtem (*xCenter*, *yCenter*) i radijusom *radius*. U igri postoji samo nekoliko mogućih veličina loptice pa je uveden atribut *sizeLevel*, o kojem ovisi radijus loptice. Najmanje loptice koje se pojavljuju u igri su one kod kojih je *sizeLevel = 1*, a predviđeno je da su najveće loptice koje se pojavljuju one kod kojih je *sizeLevel = 6*. Kad god igrač pogodi neku lopticu svojim kopljem, ona se razdvaja u dvije loptice čiji je *sizeLevel* za jedan manji. Kada igrač pogodi lopticu za koju je *sizeLevel = 1*, ona nestaje. Trenutno su *radius* i *sizeLevel* povezani tako da je *radius = sizeLevel \* 10*. Loptice također imaju brzinu i smjer kretanja određen s (*xVelocity, yVelocity*) koji se zbraja sa središtem loptice prilikom svakog pomaka u metodi *update()*. U metodi *update()* se također provjerava je li loptica udarila u neki od rubova ekrana, čime se njena brzina mijenja na odgovarajući način. Osim toga, implementirane su metode *draw()* za iscrtavanje loptice te *checkSpearCollision()* i *checkPlayerCollision()* koje primaju koordinate glave koplja, odnosno poziciju igrača te provjeravaju nalaze li se one u radijusu oko središta loptice. – 5 sati
  + U glavnom programu su napisane metode koje se brinu za pomicanje igrača, *keyPressed()* i *keyReleased()* te dvije pomoćne preopterećene metode *setMove()*. Također je dodana varijabla *state* koja se koristi za postavljanje i određivanje trenutnog stanja programa, a planirano je da bude u jednom od stanja *MAINMENU*, *SETTINGS* ili *GAME*, definiranih u enumeraciji *State*. U glavnoj *draw()* funkciji se potom iscrtavaju samo oni elementi ili pozivaju one metode koje pripadaju jednom od stanja (za sad su to samo elementi stanja *GAME*). Tako se u glavnoj *draw()* funkciji prilikom stanja *GAME* iscrtavaju loptice, igrač te se provjerava postoje li kolizije. U svrhu provjera kolizija, implementirane su metode *ballSpearCollision()* te *ballPlayerCollision()* koje prvo za sve postojeće loptice pozivaju ranije spomenute metode *Ball.checkSpearCollision()* i *Ball.checkPlayerCollision()* te obrađuju posljedice kolizije (nestajanje ili razdvajanje loptice, odnosno gubitak života). Obrada posljedica (npr. završetak igre – *Game over*) nije još u potpunosti implementirana. – 4 sata
* 10. ožujka 2021. (Dominik Mikulčić):
  + Implementirano dijeljenje lopte kad ona udari koplje bilo gdje ispod vrha (riječ je o malim promjenama u funkciji *Ball.CheckSpearCollision*), potom je prilagođeno oduzimanje života (dodavanjem atrubuta *is\_being\_hit* klasi *Ball* koji pamti udara li lopta igrača) te su zatim funkcionalnosti igre prilagođene za dva igrača (dodavanjem liste *players* koja sadrži sve igrače pa se funkcionalnosti implementiraju na sličan način kao kod liste *balls* i dodavanjem pomoćne varijable *quantity* koja pamti broj igrača) – 6 sati
* 16. ožujka 2021. (Katarina Šupe):
  + Implementacija stanja *MAINMENU*. Kreiranje glavnog izbornika. U glavnom izborniku trenutno je nacrtana pozadina kao niz horizontalnih i okomitih linija te je dodan glavni lik igrice, loptica i naziv igre. U izborniku se nalazi i pravokutnik na kojem se nalaze 4 odabira. Taj pravokutnik je pozicioniran u prozoru s obzirom na fiksne veličine *windowWidth* i *windowHeight* te je visine *totalHeight*. Nakon toga je podijeljen na 4 polja visine *fieldHeight*. Predviđeno je da igrač može birati igru s jednim igračem („1 PLAYER“) ili dva igrača („2 PLAYERS“) te može pogledati upute za igru („CONTROLS“) ili izaći iz igre („QUIT“). Trenutno se pritiskom na gumb *ENTER* pokreće igra s brojem igrača određenim varijablom *quantity*. U kod je dodan novi font, odabran da nalikuje fontu u originalnoj igri. Stil koji se primjenjivao na sadržaje unutar glavnog izbornika je 'zapakiran' u *pushStyle()* i *popStyle()*, a rotacija slike *bubbleTrouble* je obavljena pomoću *pushMatrix()* i *popMatrix()*. Kreirana je enum klasa MenuPick koja sadrži moguće odabire u izborniku, koji će se mijenjati pritiskom na gumbe *UP* (gore) ili *DOWN* (dolje) te potvrđivati pritiskom na gumb *ENTER*. U enum klasi *State* enumeracija *SETTINGS* preimenovana je u *INSTRUCTIONS* te će takvo stanje odgovarati odabiru polja „CONTROLS“ u glavnom izborniku. Dalje je potrebno implementirati mijenjanje odabira polja (npr. boju pravokutnika ili pomicanje baklji - *torch*) te s obzirom na to i razliku u odabiru igre s jednim ili dva igrača. Također, potrebno je implementirati stanje *INSTRUCTIONS*, što može biti samo slika s uputama, koja nudi povratak u glavni izbornik (npr. pritiskom na neku strelicu). Zbog logike implementacije glavnog izbornika, kreirana je metoda *createPlayers()* koja kreira listu igrača s obzirom na *quantity* te se poziva svakim novim odabirom igre u glavnom izborniku. – 6 sati
* 17. ožujka 2021. (Katarina Šupe):
* Dodana je mogućnost promjene odabira u meniju. Promjena se odvija pritiskom na strelicu gore i dolje te se tada promijeni pozadina odabranog polja. Pritiskom na gumb *ENTER* potvrdi se odabir koji bi trebao ispravno voditi na različite igre ili izaći iz igre. Trenutno još ne radi odabir *CONTROLS*. Većina implementacije se odnosila na promjenu enuma *menuPick* pomoću kojeg se mijenjao željeni odabir. – 1 sat
* 17. ožujka 2021. (Lorena Novak):
* Dodani su zvukovi na igricu – intro zvuk (*intro.mp3*) i zvukovi kod pucanja (*shooting.mp3*) i pogotka (*collision.mp3*). Zvuk na početku se reproducira dok ne završi ili dok korisnik ne pokrene igricu (ovdje moguća promjena da se zvuk ne prestane izvršavati nego da svira sve dok je *MAINMENU* upaljen). Zvuk kod pucanja se reproducira kad igrač izbaci strijelu pa sve dok ona ne nestane s ekrana ili sve dok ne pogodi lopticu. Kad pogodi lopticu, reproducira se zvuk pogotka. Svi zvukovi izrezani su iz YouTube videa skinutog u mp3 obliku.
* Dodan *soundOn* / *soundOff* gumb. Svi zvukovi se reproduciraju jedino ako je boolean *soundOn* postavljen na *true*. Pritiskom na gumb, mijenja se sličica ovisno o tome jesmo li upalili ili ugasili zvuk. Ukupno rada – 5h.
* TODO: pogledati reproduciraju li se zvukovi u skladu s igricom, srediti intro i dodati ostale zvukove.
* 18. ožujka 2021. (Lorena Novak):
* Dodani zvukovi udarca loptice i igrača i zvuk promjene opcije u glavnom izborniku (nakon pritiska tipke Enter, reproducira se zvuk)
* Popravljene koordinate *soundOn/Off* gumba
* Dodan *delay* od 0,5s ako se izgubi život. Dodan novi boolean *lostLife* koji se postavlja na *true* ako je igrač izgubio život i onda u fji *draw* provjeravamo je li *lostLife == true* i ako je, odgodimo vrijeme crtanja za pola sekunde i promijenimo *lostLife* na *false.*
* U klasi Player je dodana fja *stopSpearSound()* koja gasi zvuk strelice.

Ukupno rada: 1h.

* 18. ožujka 2021. (Dominik Mikulčić):
  + Ukoliko se izgubi život, uređen je prijelaz od pola sekunde tokom kojeg se prikazuje natpis i igra se zamrzne (loptice se prestanu kretati i onemoguće se tipke za micanje). U tu svrhu napisane su funkcija *write\_dummy\_text(String text)* koja ispisuje *text* u okvir na sredini ekrana. Također je napisana funkcija *pause\_game()* koja zaustavi loptice i onemogući tipke (osim tipke *enter*). Usto se koristi varijabla *lostLife* koja pamti je li izgubljen život – ona se postavlja na *true* u funkciji *ballPlayerCollision()* pa se u funkciji *draw()* detektira ta vrijednost i poduzima opisana akcija.
  + Jednak je prijelaz napravljen prilikom ponovnog početka igre nakon izgubljenog života. Tada koristimo varijablu *get\_ready* koja pamti nalazimo li se u fazi ponovnog početka igre. Tada se također u funkciji *draw()* pomoću funkcije *write\_dummy\_text(String text)* ispisuje prigodna poruka i čeka pola sekunde. Tada se ponovno postavljaju potrebne varijable za igru i poziva se funkcija *restart\_the\_balls()* koja pokreće loptice.
  + Ukoliko igrač izgubi sve živote, njega prestajemo crtati u funkciji *draw()*. Međutim, ako drugi igrač (u multiplayer načinu) još uvijek ima preostale živote, on nastavlja igru. Jednom kad svi igrači izgube sve živote pozivamo funkciju *game\_over()*. U njoj postavljamo varijablu *is\_game\_over* na *true* i pozivamo *pause\_game()* kako bismo zaustavili loptice i kontrole. Potom u funkciji *draw()* ispisujemo ponovno odgovarajuću poruku pomoću *write\_dummy\_text(String text)* i pomoću tipke enter se možemo vratiti na početni ekran.
  + Također, uspostavljen je bodovni sustav za igru. Temelji se na tome da najveća loptica nosi 10 bodova, a svaka sljedeća manja nosi 10 bodova više. U tu svrhu je uveden atribut *Ball.origin* koji pohranjuje polazni level loptice (npr. ako kreiramo lopticu levela 5, postavljamo i njezin *origin* na 5; kad je igrač razbije, level novonastale loptice je dakle 4, ali njezin *origin* je i dalje 5; analogno i ako se nova loptica opet razbije – i njezin je *origin* 5). Svaki put kad igrač pogodi lopticu, dakle, dobije odgovarajući broj bodova. Usto, kad je level završen, dobije se još određena količina bodova ovisno o količini vremena potrebnoj za završetak levela. Naime, za svaki ćemo level postaviti neki vremenski interval, pa ako je unutar njega dovršen level dobiju se dodatni bodovi (za svaku preostalu sekundu – 5 bodova).
  + Konačno, ako igrač pobijedi (uništi sve loptice a preostane mu bar jedan život) poziva se funkcija *levelWon()*. Postavljamo varijablu *level\_done* na *true*, pozivamo *pause\_game()* te dohvatimo trenutno vrijeme kako bismo izračunali broj ostvarenih bonus bodova. U *draw()* analogno kao u prethodnim slučajevima ispisujemo poruku i usto ostvareni broj bodova. Također, pomoću enter se vraćamo na početni ekran.
  + Ukupno 8 sati rada.
* 19. ožujka 2021. (Katarina Šupe):
* Dodan je odabir CONTROLS, tj. implementirano je stanje INSTRUCTIONS u kojem se nalazi fotografija s objašnjenjem na koji način se igrači mogu kretati i koristiti koplje u igri. Za povratak u glavni izbornik potrebno je pritisnuti gumb *MENU*. Za paljenje/gašenje zvuka dodana je nova slika. Također, napravljena je nova pozadina za igricu koja više nalikuje originalnoj. – 1 sat
* 19. ožujka 2021. (Tihana Štifanić):
  + Dodano računanje dimenzije različitih slika igrača (slika za kretanje u različitim smjerovima) tako da sve slike imaju omjer širine i visine kao izvorne datoteke. Također, slika koplja je ubačena u igru (i spremljena u varijablu *spearImg*).
  + Popravljena je greška u kojoj se igra rušila prilikom korištenja lijeve i desne strelice dok je igrač u meniju – dodana je provjera trenutnog stanja u obje metode *setMove()* tako da pritisak lijeve i desne strelice, kao i tipki A, S i D ima učinka samo onda kada je trenutno stanje GAME.
  + Popravljena je greška u kojoj je igrač mogao izaći izvan okvira igre – dodana je provjera trenutne pozicije igrača prilikom njegova pomicanja u metodi *Player.draw()* kako ona ne bi mogla postati manja, odnosno veća, od lijevog, odnosno desnog, ruba okvira igre.
  + Slika koja se koristi kao pozadina menija je dodana kao pozadina igre.
  + Ukupno 2 sata rada.
* 19. ožujka 2021. (Dominik Mikulčić):
  + Budući da je level loptice ograničen na 6, atribut *Ball.origin* nije zapravo bio potreban. Stoga njega brišemo i dodjeljujemo bodove tako da lopta levela 6 donosi 10 bodova, a svaka sljedeća manja 10 više – pa loptica levela 1 nosi 60 bodova.
  + Popravljen je glitch u kojem se prilikom gubitka života u multiplayer načinu igra zamrzne: dodali smo uvjet u *ballPlayerCollision()* na samom početku da ako je varijabla *lostLife* postavljena na *true* se izvršavanje funkcije prekida. Također smo dodavanjem novog atributa *Player.just\_lost\_life* odlučili pamtiti je li točno određeni igrač u danom trenutku izgubio život – time smo u funkciji *draw()* spriječili da se on prerano prestane iscrtavati.
  + Također, popravljeni su problemi s prikazivanjem broja preostalih života prilagođavanjem poravnanja teksta (*textAlign()*).
  + Ukupno 1 sat rada.
* 20. ožujka 2021. (Tihana Štifanić):
  + Prilagođena visina skakanja loptica: dodana su globalna polja *ballJumpHeight* te *splitBallJumpHeight* kojima je određena maksimalna visina loptice za „normalne“ loptice i one koje su tek nastale nakon razbijanja veće loptice, za svaki mogući *sizeLevel*. Na primjer, *ballJumpHeight[1]* je uobičajena maksimalna visina loptice kod koje je *sizeLevel = 1*. Nakon što se loptica kod koje je *sizeLevel = 2* razbije na dvije manje čiji je *sizeLevel = 1*, one dodatno „poskoče“ tako da njihova maksimalna visina privremeno postaje *splitBallJumpHeight[1]*. Ovo ponašanje se kontrolira atributom *split* unutar klase *Ball* koji je jednak -1 ukoliko se loptica ponaša „normalno“, a inače je njegova vrijednost jednaka *y* koordinati veće loptice (od koje je trenutna loptica nastala) u trenutku razbijanja veće loptice.
  + Implementirano je razbijanje loptice kada ona udari u gornji rub igre. U tu svrhu je dodana metoda *ballTopEdgeCollision()* koja provjerava postoji li kolizija bilo koje loptice i gornjeg ruba ekrana te, ukoliko postoji, postupa na isi način kao da je loptica razbijena kopljem (loptica se razdvaja ili nestaje te se igraču pridodaju bodovi). U tu svrhu je također napravljena mala refaktorizacija koda u kojoj je dio obrade kolizije loptice s kopljem ili gornjim rubom ekrana prebačen u metodu *splitBall(int i, Player p)* koja razdvaja/briše lopticu indeksa *i* te pridodaje bodove igraču *p*.
  + Popravljena je greška u kojoj se igra rušila prilikom korištenja tipki A, S i D prilikom igre s jednim igračem – dodana je provjera trenutnog broja igrača (*quantity*) u metodi *setMove()* tako da pritisak tipki A, S i D ima učinka samo onda kada broj igrača nije jednak 1.
  + Prilagođene su granice piksela za provjeru kolizije lopte i igrača unutar metode *Ball.CheckPlayerCollision()* tako da one odgovaraju dimenzijama novih slika igrača.
  + Ukupno 4 sata rada.