Математичка гимназија

Матурски рад

из предмета Програмирање и програмски језици

Игра Ним

Ученик

Јован Спасојевић 4b

Ментор

професор Душа Вуковић

Београд, мај 2017

Правила и историја

Ним је логичка игра која се игра између 2 играча. Састоји se од стубова на којима се налазе предмети. Број предмета између стубова не мора бити исти. Сваки потез играча који је на реду узима колико год хоће предмета,али само са једног стуба и мора се узети бар један. У зависности од типа игре која се игра победник је онај који на крају узме послењи предмет (нормална игра) или она што на крају не узима последњи предмет (тзв. *misère* игра).

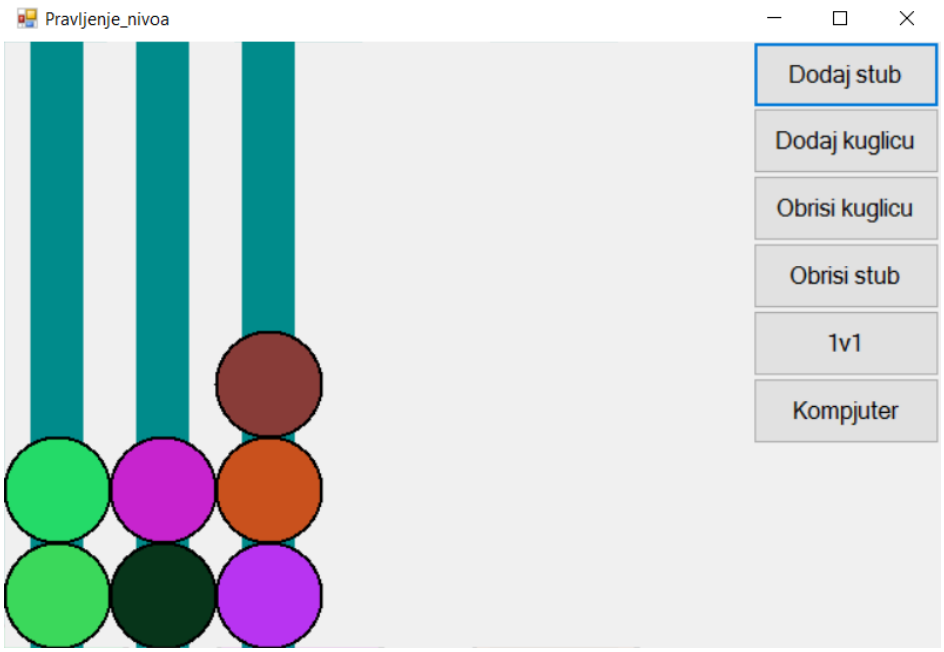
Разне варијанте Нима су се играле још од давних времена. Не зна се из које земље потиче, али се мисли да је настала у Кини зато што подсећа на њихову игру *jiǎn-shízi .* Њено данашње име јој је дао Чарлс Л. Ботон. Нормална верзија игре Ним је основа Sprague-Grundy теорије која каже да је свака нормална верзија impartial игре једнака Ним стубу који износи исту вредност када се игра паралелно са осталим нормалним верзијама те impartial игре. Impartial игра је игра где је једина разлика у потезима које могу да направе два играча јесте да први играч игра први. Користећи вредности тих стубова може се наћи стратегија за побеђивање у Impartial играма . Детаљније ће бити објашњено касније у раду.

Пројекат

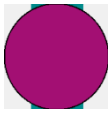
Овај пројекат се састоји из 2 фајла, ***Nim.exe*** і ***Klase.dll***. Први фајл је сачињен из 3 програма. У првом се прави ниво који ће да се игра против другог играча у другом програму или против компјутера у трећем. Други фајл је dll фајл у коме се налазе класе које се користе у главном фајлу.

Дефинисање нивоа

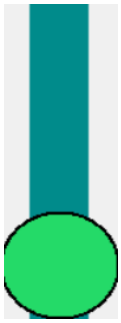
Покретањем фајла Nim.exe се отвара програм у коме се пре почетка игре дефинише ниво тежине.



У овом програму се налазе:



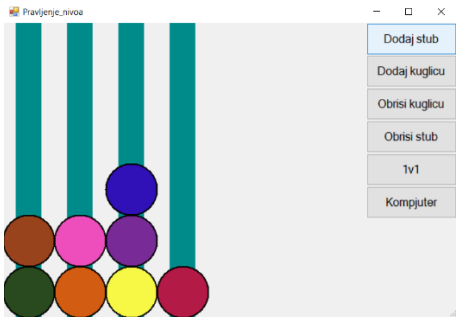
-Куглице- оне представљају предмете које ће играчи сваки потез скидати са стуба (бар једна мора да се скине по потезу и може само са једног стуба).



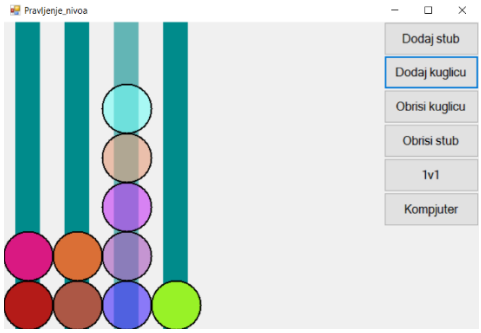
-Стубови- сваки стуб има свој засебан број куглица. Играчи ће на сваком свом потезу бирати један од више преосталих стубова са ког ће скидати куглице. (Ако стуб нема више куглица онда се брише).

На почетку се програм учита са три стуба који садрже 2,2,3 куглице. Такође, играч може да бира следеће:

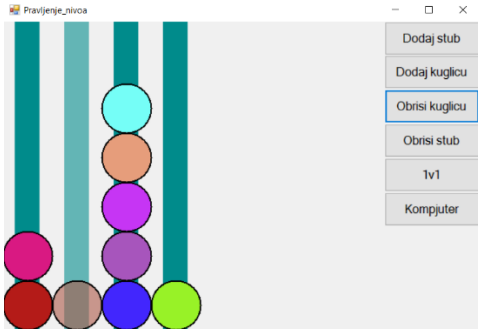
Dodaj stub: додаје један стуб са једном куглицом.



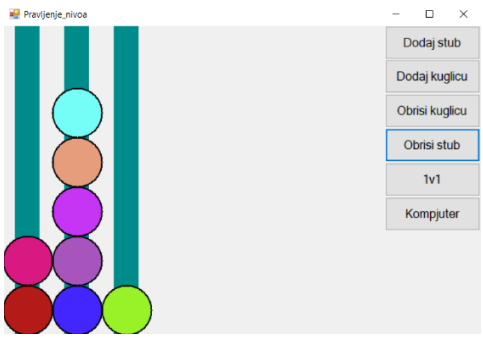
Dodaj kuglicu: додаје једну куглица на стуб.



Obrisi kuglicu: смањује број куглица на једном стубу за 1.



Obrisi stub: брише стуб и све куглице које су се налазиле на њему.



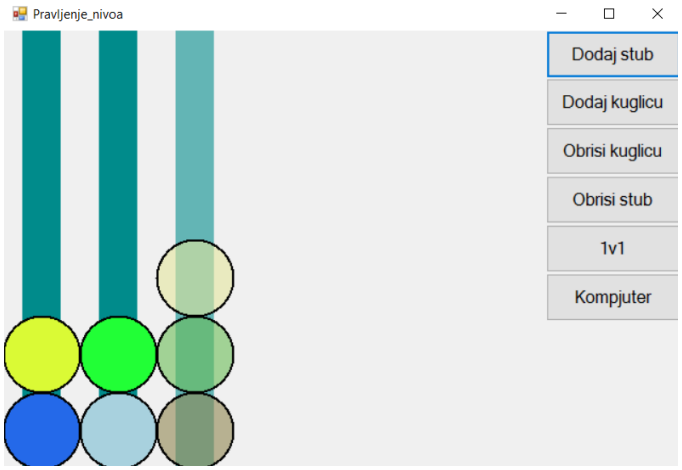
1v1: отвара програм где ће играти против другог играча.

Компјутер: отвара програм где ће се играти против компјутера. Максимални број стубова који програм подржава је 20 и број стубова се не може смањити на мање од 3. Ако се брисањем стуба укупан број куглица у програму смањи на мање од 7, стуб се неће обрисати.

Максимални број куглица који један стуб може да подржи је 20, а смањивањем се не може смањити на мање од 1 и не може се обрисати куглица уколико би се њеним смањивањем укупан број куглица смањио на мање од 7. Корисник такође може смањити или повећати број куглица користећи ***Mouse wheel*** (померање унапред=повећавање,померање уназад=смањивање).

Да би ***Obrisi stub, Dodaj kuglicu*** и ***Obrisi kuglicu*** функције биле омогућене, пре клика на дугме прво треба да се означи стуб.

Стуб се означава тако што се отпусти миш у околини стуба који треба бити изабран.

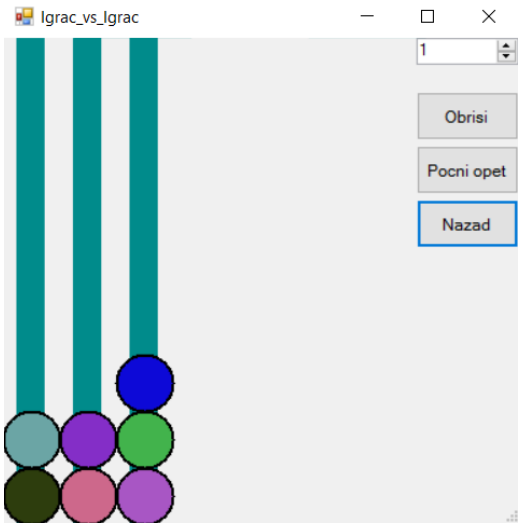


Када је стуб изабран он постаје блед као и све куглице које се налазе на њему. Бирањем другог стуба или отпуштањем миша на празном простору се стуб враћа у нормално стање. Када изабран стуб бива обрисан мора се опет кликнути на нови стуб да би се он изабрао.

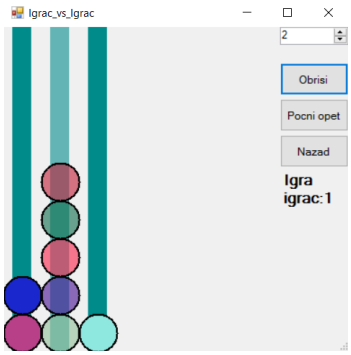
Игра

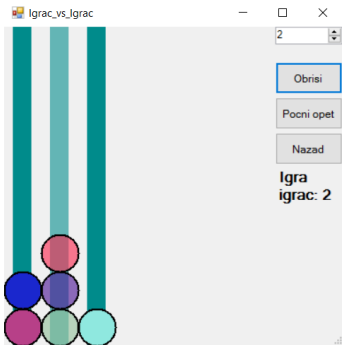
Играч против играча

Када је корисник задовољан нивоом који је направио треба да кликне на **1v1** ако жели да игра против другог играча или на **Компјутер** дугме ако жели да игра против компјутера.



Приликом отварања игре у 2 играча, учитава се ниво који смо у претходном програму направили, бројач и 3 дугмета:
Obrisi: Кликом на ово дугме се брише број куглица,одређено бројачем у горњем десном углу, означеног стуба.

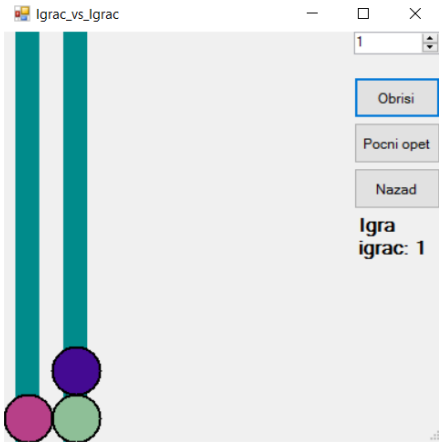


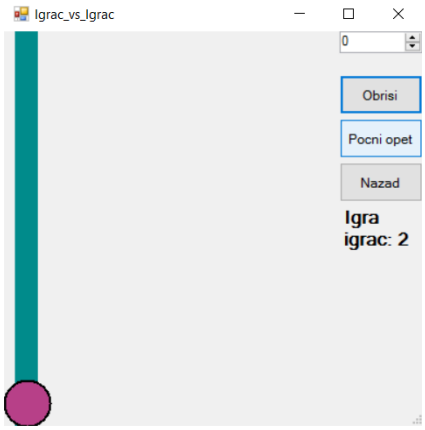


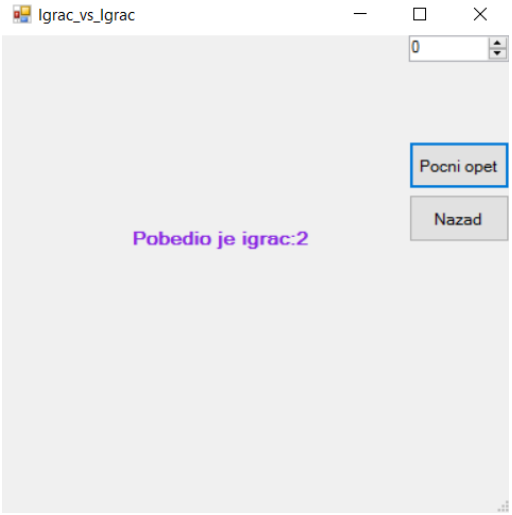
***Pocni opet*:** Ово дугме покреће игру од почетка.

***Nazad*:** Кликом на ово дугме се враћа на програм за прављење нивоа.

Играчи наизменично играју потезе кои је састоје од бирање стуба, одређивања броја куглица који желе да скине коришћењем бројача и кликом на дугме *Obrisi.* Играч који успе да скине све куглице са јединог преосталог стуба побеђује.

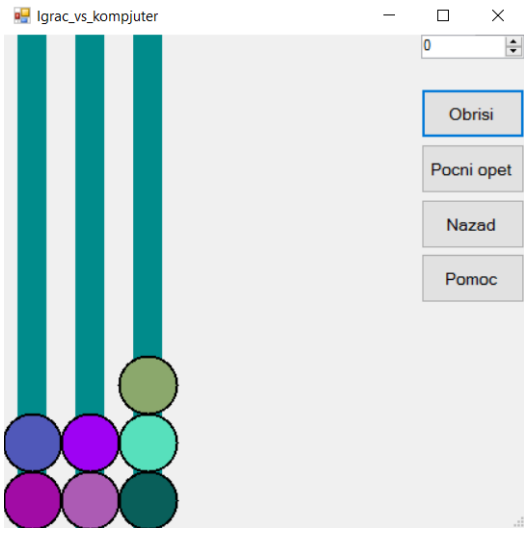






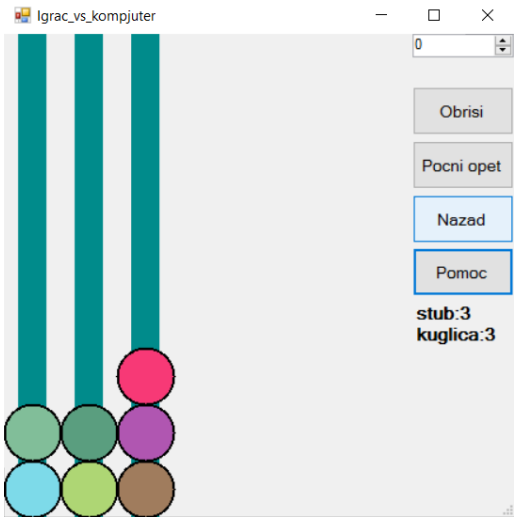
У примеру изнад први играч узима 2 куглице са другог стуба остављајући један стуб са једном куглицом. Други играч узима једину и последњу куглицу чинећи га победником игре у овом случају.

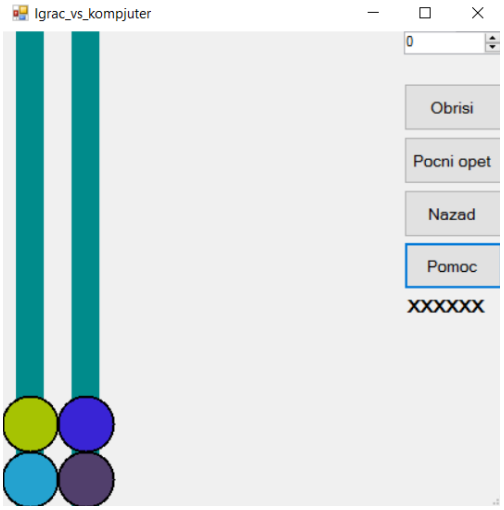
Игра против компјутера



У игри против компјутера такође имамо бројач, 3 дугмета: ***Obrisi, Pocni opet, Nazad*** , али имамо и додатно дугме ***Pomoc.***

Дугме ***Pomoc*** нам каже који потез да одиграмо,ако је могуће , како не би изгубили од компјутера и компјутер убацили у губитнички положај.





Наравно то не значи да после коришћења помоћи можемо да правимо било који потез. Морамо обазриво играти све време како не би компјутеру дали назад вођство. ***Pomoc*** дугме можемо да користимо највише 3 пута и онда ће нестати.

Код Klase.dll

У фајлу Klase.dll се налазе 3 класе које се користе у главном програму.
То су **Stub,Predmet,Min_max**.

Класа Предмет

Предмет је класа која садржи променљиве које чувају локацију,величину и боју куглице, као и методе за цртање куглице,да постане светлија итд. Класа садржи 4 променљиве.

```
private int radijus,x,y;

private Color boja;
```

Radijus променљива представља полупречник куглице, **x** и **y** представљају x и y кординате куглице и **boja** представља боју куглице.
Оне своје вредности добијају када се креира нови Предмет.

```
public Predmet(int x, int y, int radijus, Color boja)
        {
            this.x = x;
            this.y = y;
            this.radijus = radijus;
            this.boja = boja;
        }
```

У методи **Napravi** се црта куглица коришћењем `Graphics g` из форме одакле се позива.

```
public void Napravi(Graphics g)
{
 SolidBrush cetka = new SolidBrush(boja);
 g.FillEllipse(cetka, x - radijus / 2, y - radijus, 2 * radijus, 2 * radijus);
 Pen olovka = new Pen(Color.Black, 2);
 g.DrawEllipse(olovka, x - radijus / 2, y - radijus, 2 * radijus, 2 * radijus);
}
```

У методи **Svetliji** боја куглица постаје светлија и она се заједно са методом **Svetliji_Predmeti** у **Stub**-у користи за означавање кликнутог стуба, док **Normalna_boja** враћа боју куглице назад на нормално.

```
public void Svetliji()
{
  this.boja = Color.FromArgb(150, this.boja.R, this.boja.G, this.boja.B);

}
public void Normalna_boja()
{
 this.boja = Color.FromArgb(255, this.boja.R, this.boja.G, this.boja.B);
}
```

Метода **update** се користи да се приликом промене величина прозора програма промени и величина **radijus** и опет нацрта куглицу.

```
public void update(Graphics g, int radijus)
   {
       this.radijus = radijus;
       this.Napravi(g);

   }
```

Имамо и 2 методе **X,Y** за **x,y** које служе да се читају вредносту **x**,**y** или уписује нова вредност у у њих.

```
public int X { get { return x; } set { x = value; } }
public int Y { get { return y; } set { y = value; } }
```

Класа Stub

Ова класа садржи 6 променљивих које одређују његову дужину,ширину,бројпредмета,његову локацију,боју и листу предмета које тај стуб садржи.

```
private int duzina, sirina, brojpredmeta,x;
private Color boja;
public List<Predmet> lista;
```

Ове вредности добијају своје вредности током креирања новог стуба.

```
public Stub(int x, int brojpredmeta, int sirina, int duzina)
{
 this.x = x;
 this.duzina = duzina;
 this.sirina = sirina;
 this.brojpredmeta = brojpredmeta;
 this.boja = Color.DarkCyan;
 this.lista = new List<Predmet>(brojpredmeta);
}
```

Метода **NapraviPredmete** се користи да би се само креирали и задале променљиве ,не нацртале, куглице који се налазе на стубу, а `Random r` се користи се да би случајним избором изабрала боја куглице.

```
public void NapraviPredmete(Random r)
{
 for (int i = 0; i < brojpredmeta; i++)
  {
 lista.Add(new Predmet(X - sirina, duzina - (2 * i + 1) * sirina, sirina,
Color.FromArgb(255, r.Next(256), r.Next(256), r.Next(256))));
  }
}
```

Методом **Dodaj** се креира и задају променљиве једној новој куглици стуба,док се методом **Smanji** смањује број куглица за број који је дат уз методу.

```
public void Dodaj(Random r)
{
lista.Add(new Predmet(X - sirina, duzina - (2 * this.brojpredmeta + 1) * sirina,
sirina, Color.FromArgb(255, r.Next(256), r.Next(256), r.Next(256))));
this.brojpredmeta++;
public void Smanji(int broj)
{
 while (broj > 0)
 {
  this.lista.Remove(lista[lista.Count - 1]);
  this.brojpredmeta--;
  broj--;
  }
}
```

Коришћењем **Svetliji_Predmeti** се стуб и све куглице на њему осветле,а **Normalni_Predmeti** враћа све на своју првобитну боју.

```
public void Svetliji_Predmeti()
{
 for (int i = 0; i < this.BrojPredmeta(); i++)
   {
    lista[i].Svetliji();
   }
 this.boja = Color.FromArgb(150, boja.R, boja.G, boja.B);
 }

public void Normalni_Predmeti()
{
  for (int i = 0; i < this.BrojPredmeta(); i++)
   {
    lista[i].Normalna_boja();
   }
  this.boja = Color.FromArgb(255, boja.R, boja.G, boja.B);
}
```

Као и Предмет, ова класа садржи које се користи да се приликом промене величине прозора у ком се игра, промени и величина стуба као и све његове куглице.

```
public void update(Graphics g, int novx, int sirina, int duzina)
{
 this.X = novx;
 this.duzina = duzina;
 this.sirina = sirina;
 SolidBrush cetka = new SolidBrush(boja);
 g.FillRectangle(cetka, X - sirina, 0, sirina, duzina);
 for (int i = 0; i < this.BrojPredmeta(); i++)
 {
  lista[i].X = novx - sirina;
  lista[i].Y = duzina - (2 * i + 1) * sirina;
  lista[i].update(g, sirina);
  }
 }
```

Методе **BrojPredmeta** се користе да се врати број куглица стуба.

```
public int BrojPredmeta()
{
  return brojpredmeta;
}
```

X користимо да вратимо или променимо позицију стуба.

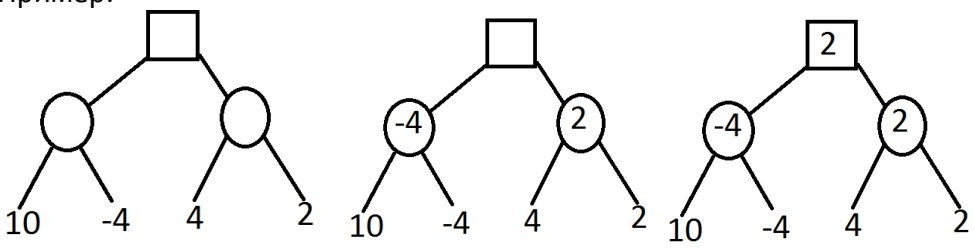
```
public int X { get { return x; } set { x = value; } }
```

Класа Min_max

Min_max алгоритам

Пре објашњења кода прво ће бити објашњено како ради алгоритам **MinMax**. **MinMax** је алгоритам који се користи у теорији игара и обично за игре са 2 играча како би минимизирао вероватноћу да изгубиш игру. Он пролази кроз све потезе оба играча како би израчунао минималну вредност поена који је први играч засигуран да добије у игри,не знајући потезе другог играча, а оба играча играју паметно (тј. не праве потезе због којих ће да изгубе). Алгоритам се решава преко рекурзије. Од почетног стања игралишта се траже сви могући потези. Игралиште после направљеног потеза ће се звати дечји чвор,а игралиште пре потеза ће се назвати родитељски чвор. После се од тих дечјих чворова праве даље сви могући дечји чворови,од тих новонасталих чворова се даље праве нови чворови итд. док се не дође до стања игралишта које ће имати неку вредност поена за првог играча. Потом се вредности свих тих чворова пореде и у зависности који је играч на потезу ,родитељском чвору ће се придати максимална вредност ако је први играч на потезу односно минимална ако је други на потезу. Тим враћањем на крају добијемо минималну вредност поена које први играч може добити ако оба играча играју паметно.

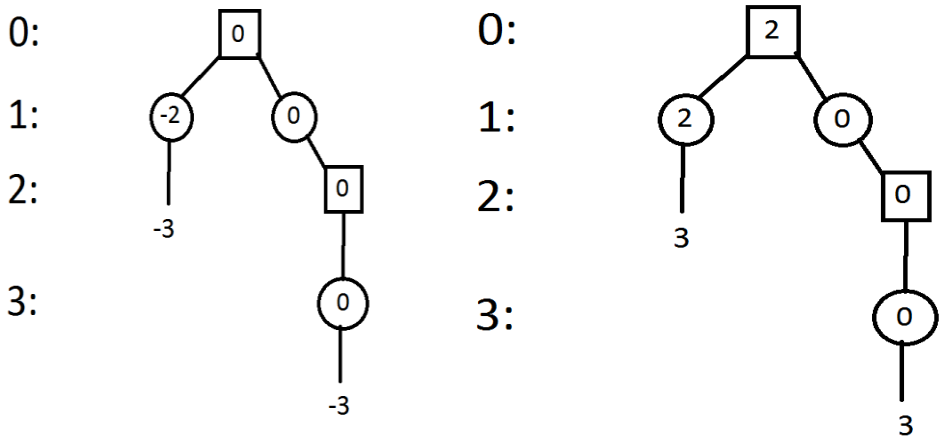
Пример:



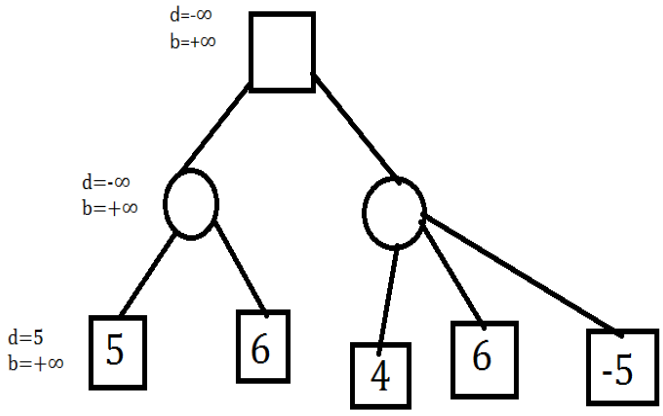
У горњем примеру **коцка** представља првог играча,а **круг** другог. Пошто други играч игра тако да првом зада што мање поена, у левој грани ће изабрати -4, а у десној 2. Први играч игра тако да добије што више поена па ће он изабрати 2 па је минималан број поена што први играч добија не знајући противничке потезе је 2.

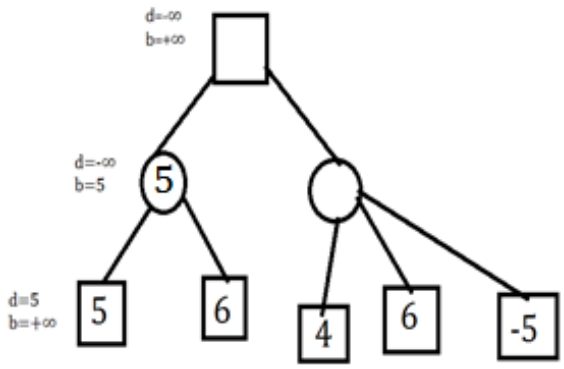
Овим примером је показана важност **MinMax** алгоритма јер иако левим гранањем први играч би могао да освоји 10 поена, такође исто тако може да добије -4 поена. Зато је најсигурније ићи десним гранањем зато што је минимум који ту може да се освоји 2 поена.

Даље у овом програму је коришћено 2 начина побољшања **MinMax** алгоритма. Први начин је додавањем **нивоа**. З Сваки потез се означава са неким нивоом. На пример почетни стање игралишта је нулти ниво,први потез је први ниво,други потез други ниво итд. За ралзику од прошлог примера,неће се свако гранање завршити на истом нивоу. Па се ови нивои користе да у случају да имамо 2 пута до минималног броја поена првог играча,пут са најмањим бројем грана се бира ако је победио односно пут са највишим бројем грана се бира ако је изгубио. (Поени мањи од 0 значе да је први играч **изгубио**,а бројеви већи од 0 и 0 да је **победио**). У случају победе се на број поена дода број нивоа на ком се налази крај гранања,а у супротном случају се од броја поена одузме број нивоа.

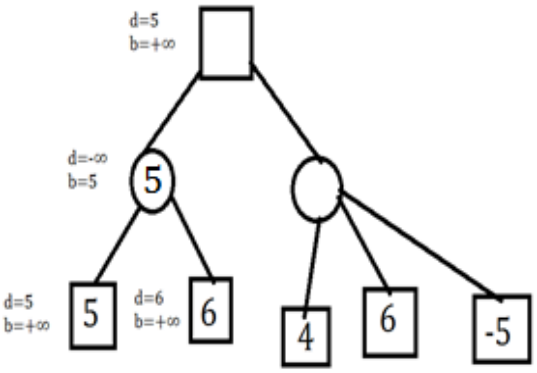


Други начин је коришћењем граница. Он нам служи да смањимо број пролаза кроз чворове тако што неке гране нећемо гледати јер неће дати боље решење. Сваки чвор ће имати 2 променљиве **алфа** и **бета**. На почетку главни чвор ће имати **алфа**=-бесконачно(доња граница)и **бета**=+бесконачно(горња граница.). Те вредности даље наслеђују дечји чворови и тако до краја док се не дође до краја гранања. Потом се проверава који ке играч на потезу. Алфа добија вредност поена ако је први играч на потезу односно бета добија вредност поена ако је други на потезу.

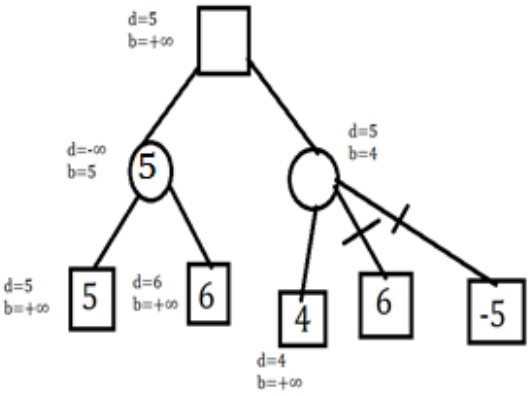




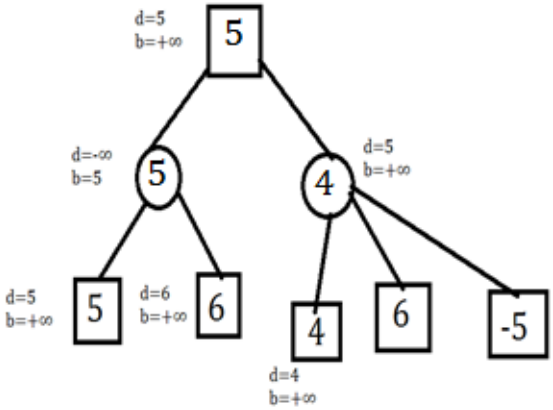
Бета вредност родитељског чвора добија вредност алфа од дечијег ако је алфа мања од бета(родитељски чвор је потез другог играча) односно алфа вредност родитељског чвора добија вредност бете дечијег ако је бета веће од алфа(родитељски чвор је потез првог играча).



Те новодобијене алфа односно бета родитељског чвора се потом пореде са осталим дечјим чворовима и мењају вреднист родитељског ако могу. У овом примеру ново родитељско бета=5 је мање од дечијег алфа=6 па се бета неће променити.



Главни родитељски чвор има нову вредност алфа=5. За свако наредно гранање ће дечији чворови наследити те промењене вредности.



На крају ће се десити да родитељски чвор десног гранања имати вредности алфа=5 и бета=4. То говори да се даље гранање не мора гледати јер се неће добити бољи минимум.

Алфа главног чвора после завршетка алгоритма ће представљати минимум број поена који први играч добија ако игра паметно.

Разлог зашто ово ради је иако у десном гранању постоји дечји чвор са вредношћу 6 он неће никад бити изабран јер други играч бира најмању могућу вредност коју може да наметне првом играчу, а пошто је прво изабрано 4 добијамо да ће десна грана дати мањи минимум него лева па се даље гранање не мора гледати.

Одређивање крај гранања и рачунање поена

Крај гранања се рачуна на 3 начина.

1.Остао је један стуб

Ако је остао само један стуб, играч који је тренутно на потезу само треба да узме све куглице са стуба да би победио. Вредност краја гранања ће бити:

Победник 1. играч=100-број нивоа.
Победник 2.играч=-100+број нивоа.

2.Ним-сабирање(+*)

На почетку рада је речено да постоји постоји победничка стратегија која користи вредност стубова. Сада ће бити објашњена.
Сваки стуб ће добити вредност ***k** где **k** представља број куглица тог стуба.Те вредности ће се звати ним-вредности. Приликом сабирања тих ним-вредности ако им је збир 0 то значи да други играч побеђује који год потез да прави.
Када имамо 2 стуба са истим бројем куглица први играч је изгубио који год потез да направи зато што колико год куглица да скине са једног стуба,други играч само треба да скине исти број куглица са другог. Сабирање тих ним-вредности се записује као ***k+*k=0** тј ***k=-*k** за било које к>1. Од овог примера следи да је ним-вредност једнака својој негативној вредности и ако има само 2 стуба, играч који први изједначи оба ће увек победити тј.
k+*n-*(n-k)=k+*(k+n-k) -*(n-k)=*k+*k+*(n-k) -*(n-k)=0 (користи се ***k=-*k** и ***k+*k=0),n>k**
Збир 2 ним-вредности је нова ним-вредност:
k+*n=k+*(k+n-k) +*(n-k)=*k+*k+*(n-k)=*(n-k),n>k.
Коришћењем ***k+*k=0** се може се одредити да ли игра којас се игра може победити. Ним вредности ће се сада записивати у облику $*2^b+*a=*k$.
Пример:$*1+*2+*3=*2^0+*0+*2^1+*0+*2^1+*1=(*1+*1)+(*2^1+*2^1)=0+0=0$
Овај пример нам каже да када имамо 3 стуба са 1,2,3 куглице, за било који потез првог играча други играч може да направи потез који ће га довести до победе.
Ово сабирање можемо да гледамо као и да декадно сабирамо бинарне вредности ових бројева ,а случају када добијамо број 2 то гледамо као да је 0.
$1_{10}=1_2$, $2_{10}=10_2$,$3_{10}=11_2$ $*1+*10+*11=*11+*11=*22=*00=0$.

Када одредимо да је збир ним-вредности стубова 0 ,неког стања игралишта, онда се враћа:
Први играч на потезу=-100+број нивоа.
Други играч на потезу=100-број нивоа.

3. Достигнута је максимална дубина

MinMax алгоритам ,ако има пуно гранања, може да ради чак и неколико минута. Да би се скратио број тог гранања у зависности од укупног броја куглице у целој игри на почетку игре , одређује се колико дубоко може алгоритам да се грана. Вредности које враћа овај крај су небитна јер помоћу прва 2 начина ће алгоритам наћи потезе којим ће компјутер играти. Овај начин само служи да пренике даље гранање.

Први играч на потезу=-50+број нивоа.
Други играч на потезу=50-број нивоа.

Код класе MinMax

Ова класа садржи од променљива: 2 public int-а који ће после завршетка алгоритма имати вредности потеза који компјутер треба да одигра, један што проверава `bool` да ли се нешто десило и `Random R` који се користи да се од од 2 краја гранања са истим вредношћу на истој дубини,избор потеза бира случајно.
Такође садржи листу,чији су чланови укупан број куглица једног стуба, која показује стање терена после направљеног потеза.

```
List<int> polje;
private int dubina_max;
public int broj_stuba,broj_kuglica;
private bool provera;
Random R = new Random();
```

Код испод се одиграва када се направи нови **MinMax**

```
public Min_max(int dubina_max)
 {
  this.dubina_max = dubina_max;
  provera = true;
  broj_stuba = 0;
  broj_kuglica = 0;
}
```

Binary се користи да би се број превео у своју бинарну вредност

```
 private int Binary(int broj)
{
 return Convert.ToInt32(Convert.ToString(broj, 2));
}
```

Метода Zbir декадно сабира 2 бинарна броја и у случају да у збиру постоје двојке оне ће се заменити са 0.

```
private int Zbir(int broj1, int broj2)

{
 string broj = (broj1 + broj2).ToString();
 string res = "";
 for (int i = 0; i < broj.Length; i++)
  {

   if (broj[i] == '2')
    {
     res += "0";
    }
   else
    res += broj[i];
   }
  return Convert.ToInt32(res);
 }
```

Provera-ом се одређује да ли је тренутно стање игре може да врати неку вредност.

```
private bool Provera(List<int> brojpredmeta)
{
 int x = Binary(brojpredmeta[0]);
 for (int i = 1; i < brojpredmeta.Count; i++)
  {
   x = Zbir(x, Binary(brojpredmeta[i]));
  }
 if (x == 0)
  return true;
 return false;

}
```

Даљи код ради мање-више што је објашњено у начину рада Minmax алгоритма са додатком bool provera који проверава да ли је на дубини 1 једног гранања алфа веће од бета. Ако јесте онда тај потез не разматрамо као један од победничких потеза. Int vrednost представља вредност која ће се рекурзијом враћати. У случају када се добија да компјутер не може да направи победнички потез, ради се проверabroj_kuglica>=j да би се проверило да ли потез који покушава да направи продужује игру.

```
public int Minmax(bool max, List<int> brojpredmeta, int dubina, int alfa, int beta)
 {
  if (brojpredmeta.Count == 1)
   {
    if (dubina == 0)
     {
      broj_stuba = 0;
      broj_kuglica = brojpredmeta.Count();
     }
    if (max)
     return 100 - dubina;
    else return -100 + dubina;
   }
  else if (dubina!=0 && Provera(brojpredmeta) )
   {
    if (max)
     return -100 + dubina;
    else return 100 - dubina;
   }
  else if (dubina == dubina_max)
   {
    if (max)
     return 50 - dubina;
    else return -50 + dubina;
   }
  else if (max)
```

```
   {
    for (int i = 0; i < brojpredmeta.Count(); i++)
   {
    for (int j = 1; j <= brojpredmeta[i]; j++)
  {
    polje = new List<int>();
   for (int z = 0; z < brojpredmeta.Count(); z++)
    {
     polje.Add(brojpredmeta[z]);
    }
   polje[i] -= j;
   if (polje[i] == 0)
    polje.RemoveAt(i);
   int vrednost = Minmax(false, polje, dubina + 1, alfa, beta);
   if (dubina == 0 && provera)
    {
     if (alfa < vrednost)
      {
       broj_stuba = i;
       broj_kuglica = j;
      }
     else if (alfa == vrednost)
      {
       if (vrednost > 0)
        {
         if (R.Next(0, 2) == 0)
          {
           broj_stuba = i;
           broj_kuglica = j;
        }
       }
    else
     if (R.Next(0, 2) == 0 && broj_kuglica>=j)
      {
       broj_stuba = i;
       broj_kuglica = j;
      }
    }
   }
 if(dubina==0)
 provera = true;
 alfa = Math.Max(alfa, vrednost);
 if (alfa >beta)
 {
  return beta;
 }
 }
}
return alfa;
}
else
{
for (int i = 0; i < brojpredmeta.Count(); i++)
{
for (int j = 1; j <= brojpredmeta[i]; j++)
{
 polje = new List<int>();
for (int z = 0; z < brojpredmeta.Count(); z++)
 {
  polje.Add(brojpredmeta[z]);
 }
 polje[i] -= j;
 if (polje[i] == 0)
 polje.RemoveAt(i);
 int vrednost = Minmax(true, polje, dubina + 1, alfa, beta);
 beta = Math.Min(beta, vrednost);
 if (beta < alfa)
  {
   if (dubina == 1)
    provera = false;
    return alfa;
  }
 }
}
  return beta;
 }
}
```

Код Nim.exe

Pravljenje_nivoa

Главни програм има разне променљиве:
3 int-а који представљају број стубова,кликнут стуб и радијус куглица;
2 листе-једна за Stub-ове ,друга за укупан број куглица по стубу која ће се користити у другим деловима програма;
`Random r` која се користи за случајан избор боје куглице;

`bool poruka` Која се користи да провери да ли је грешка изашла;
`odgovor` -представља одговор који је изабран да се затвори прозор грешке;

```
int brojstubova, izabran_stub, radijus;
List<Stub> stubovi;
Random r;
public static List<int> brojpredmeta;
bool poruka = false;
DialogResult odgovor;
```

На учитавања програма се направе 3 стуба који имају 2,2,3 куглице.

```
private void Pravljenje_nivoa_Load(object sender, EventArgs e)
 {
  brojstubova = 3;
  stubovi = new List<Stub>(10);
  brojpredmeta = new List<int>(10);
  r = new Random();
  radijus = (int)((0.8) * ClientRectangle.Width / 14);
  for (int i = 0; i < brojstubova; i++)
    {
   if (i == 0)
    {
     stubovi.Add(new Stub(i * 2 * radijus + 3 * radijus / 2,2, radijus,
ClientRectangle.Height));
     stubovi[i].NapraviPredmete(r);
     brojpredmeta.Add(2);
    }
   else
    {
     stubovi.Add(new Stub(i * 2 * radijus + 3 * radijus / 2, i + 1, radijus,
ClientRectangle.Height));
     stubovi[i].NapraviPredmete(r);
     brojpredmeta.Add(i + 1);
    }
    }
  izabran_stub = 0;
}
```

Стубови и куглице се цртају у Paintu-у. Такође се ту позива метода која мења позиције дугмића при промени величине прозора. Метода Provera се користи да се провери да ли ће кулгице при цртању прећи задате границе и смањи величину радијуса ако прећу.

```
private void Pravljenje_nivoa_Paint(object sender, PaintEventArgs e)
{
```

```
 radijus = (int)((0.8) * ClientRectangle.Width / 14);
 Provera();
 for (int i = 0; i < brojstubova; i++)
  {
   stubovi[i].update(e.Graphics, i * 2 * radijus + 3 * radijus / 2, radijus,
ClientRectangle.Height);
  }
   Dugmeupdate(button1, new Point(4 * ClientRectangle.Width / 5, 0), new
Size(ClientRectangle.Width / 5, ClientRectangle.Height / 9));
   Dugmeupdate(button2, new Point(4 * ClientRectangle.Width / 5, ClientRectangle.Height / 9),
new Size(ClientRectangle.Width / 5, ClientRectangle.Height / 9));
   Dugmeupdate(button3, new Point(4 * ClientRectangle.Width / 5, 2*ClientRectangle.Height / 9),
new Size(ClientRectangle.Width / 5, ClientRectangle.Height / 9));
   Dugmeupdate(button4, new Point(4 * ClientRectangle.Width / 5, 3*ClientRectangle.Height / 9),
new Size(ClientRectangle.Width / 5, ClientRectangle.Height / 9));
   Dugmeupdate(button5, new Point(4 * ClientRectangle.Width / 5, 4*ClientRectangle.Height / 9),
new    Size(ClientRectangle.Width / 5, ClientRectangle.Height / 9));
   Dugmeupdate(button6, new Point(4 * ClientRectangle.Width / 5, 5 * ClientRectangle.Height /
9), new Size(ClientRectangle.Width / 5, ClientRectangle.Height / 9));
  if (odgovor == DialogResult.OK || odgovor == DialogResult.Cancel)
  {
  odgovor = 0;
  poruka = false;
  }
}
private void Dugmeupdate(Button dugme, Point point, Size size)
        {
            dugme.Location = point;
            dugme.Size = size;
            dugme.Font = new Font(Font.FontFamily, (ClientRectangle.Height / 42 + ClientRectangle.Width / 42)
/ 2);

        }
private void Provera()
 {
  if (0.8 * ClientRectangle.Height <= brojpredmeta.Max() * 2 * radijus)
  {
   radijus = (int)((0.8) * ClientRectangle.Height / brojpredmeta.Max() / 2);

  }
  if (0.8 * ClientRectangle.Width <= brojstubova * 2 * radijus)
  {
   radijus = (int)((0.8) * ClientRectangle.Width / (brojstubova) / 2);

  }
}
private void Pravljenje_nivoa_Resize(object sender, EventArgs e)
{
 Refresh();
 this.MinimumSize = new Size(400, 400);
}
```

Овај програм има 6 дугмића који чија је радња објашњена на почетку:
button1=Dodaj stub;
button2=Dodaj kuglicu;
button3=Obrisi kuglicu;
button4=Obrisi stub;
button5=1v1;
button6=Kompjuter

Додаје се стуб са једном куглицом ако је могуће.

```
private void button1_Click(object sender, EventArgs e)
        {
            if (brojstubova < 20)
            {
                stubovi.Add(new Stub(brojstubova * 2 * radijus + 3 * radijus / 2, 1, radijus,
ClientRectangle.Height));
                stubovi[brojstubova].NapraviPredmete(r);
                brojstubova++;
```

```
            brojpredmeta.Add(1);
            Refresh();
        }
        else
        {
            poruka = true;
            odgovor=MessageBox.Show("Ne moze vise stubova");
        }
    }
```

Додаје се једна куглица на стуб ако је могуће.

```
    private void button2_Click(object sender, EventArgs e)
    {
        if (izabran_stub == 0)
        {
            poruka = true;
            odgovor = MessageBox.Show("Nisi izabrao stub");
        }
        else if (stubovi[izabran_stub - 1].BrojPredmeta() < 20)
        {
            stubovi[izabran_stub - 1].Dodaj(r);
            brojpredmeta[izabran_stub - 1]++;
            stubovi[izabran_stub - 1].lista[stubovi[izabran_stub - 1].BrojPredmeta() - 1].Svetliji();
            Refresh();
        }
        else
        {
            poruka = true;
           odgovor= MessageBox.Show("Ne moze vise kuglica da se doda");
        }
    }
```

Брише једну куглица са стуба ако је могуће.

```
    private void button3_Click(object sender, EventArgs e)
    {
        int broj = 0;
        for (int i = 0; i < brojstubova; i++)
        {
            broj += stubovi[i].BrojPredmeta();
        }

        if (izabran_stub == 0)
        {
            poruka = true;
            odgovor = MessageBox.Show("Nisi izabrao stub");
        }
        else if (broj < 8)
        {
            poruka = true;
           odgovor= MessageBox.Show("Minimum broj kuglica je 7");
        }
        else if (stubovi[izabran_stub - 1].BrojPredmeta() == 1)
        {
            poruka = true;
            odgovor=MessageBox.Show("Ne moze vise kuglica da se skine sa stuba");
        }
        else
        {
            stubovi[izabran_stub - 1].Smanji(1);
            brojpredmeta[izabran_stub - 1]--;
            Refresh();
        }
    }
```

Брише цео стуб са куглицама ако је могуће.

```
    private void button4_Click(object sender, EventArgs e)
    {
        int broj = 0;
        for (int i = 0; i < brojstubova; i++)
        {
            broj += stubovi[i].BrojPredmeta();

        }
        if (izabran_stub == 0)
        {
            poruka = true;
            odgovor = MessageBox.Show("Nisi izabrao stub");
        }
        else if (brojstubova == 3)
        {
            poruka = true;
            odgovor = MessageBox.Show("Ne moze stub da se obrise. Minimum broj stubova je 3");
        }
        else if (izabran_stub - 1 == -1)
        {
```

```
                poruka = true;
                odgovor = MessageBox.Show("Nisi izabrao stub");
            }
            else if (broj - stubovi[izabran_stub - 1].BrojPredmeta() < 7)
            {
                poruka = true;
                odgovor=MessageBox.Show("Ne moze stub da se oduzme zato sto  bi onda preostali broj kuglica bio manji
od 7");
            }
            else
            {
                stubovi.Remove(stubovi[izabran_stub - 1]);
                brojstubova--;
                brojpredmeta.RemoveAt(izabran_stub - 1);

                izabran_stub = 0;
                Refresh();

            }
        }
```

Започиње игру између два играча са тереном који је направљен у овом програму.

```
        private void button5_Click(object sender, EventArgs e)
        {

                this.Hide();
                Igrac_vs_Igrac nova_igra = new Igrac_vs_Igrac();
                nova_igra.ShowDialog();
                this.Close();

        }
```

Започиње игру између играча и компјутера са тереном који је направљен у овом програму.

```
        private void button6_Click(object sender, EventArgs e)
        {
            this.Hide();
            Igrac_vs_kompjuter nova_igra = new Igrac_vs_kompjuter();
            nova_igra.ShowDialog();
            this.Close();
        }
```

Такође постоји `MouseWheel` догађај који при померању точкића на мишу може да се дода или смањи број куглица `ако је дозвољено.`

```
private void Pravljenje_nivoa_MouseWheel(object sender, MouseEventArgs e)
        {
            int broj = 0;
            for (int i = 0; i < brojstubova; i++)
            {
                broj += stubovi[i].BrojPredmeta();
            }
            if (!poruka)
            {
                if (izabran_stub == 0)
                {
                    poruka = true;
                    odgovor = MessageBox.Show("Nisi izabrao stub");
                }
                else
                {
                    if (e.Delta > 0)
                    {
                        if (stubovi[izabran_stub - 1].BrojPredmeta() < 20)
                        {
                            stubovi[izabran_stub - 1].Dodaj(r);
                            brojpredmeta[izabran_stub - 1]++;
                            stubovi[izabran_stub - 1].lista[stubovi[izabran_stub - 1].BrojPredmeta() -
1].Svetliji();

                            Refresh();
                        }
                        else
                        {
                            poruka = true;
                            odgovor = MessageBox.Show("Ne moze vise kuglica da se doda");
                        }
                    }
                    else
                         if (broj < 8)
                    {
                        poruka = true;
                        odgovor = MessageBox.Show("Minimum broj kuglica je 7");
                    }
```

```
                else if (stubovi[izabran_stub - 1].BrojPredmeta() == 1)
                {
                    poruka = true;
                    odgovor = MessageBox.Show("Ne moze vise kuglica da se skine sa stuba");
                }
                else
                {
                    stubovi[izabran_stub - 1].Smanji(1);
                    brojpredmeta[izabran_stub - 1]--;
                    Refresh();
                }
            }
        }
    }
```

Догађај MouseUp бира стуб на ком смо одпустили миш и изабран стуб ће се нацртати осветљено, ако се отпусти миш ван простора стубова неће бити изабран стуб и сви стубови ће се нацртати нормално.

```
private void Pravljenje_nivoa_MouseUp(object sender, MouseEventArgs e)

    {
        int broj = 0;
        if(izabran_stub>0)
        stubovi[izabran_stub-1].Normalni_Predmeti();
        izabran_stub = 0;
        Refresh();
        while (broj < brojstubova)
        {
            if (e.X < stubovi[broj].X + radijus / 2)
            {
                stubovi[broj].Svetliji_Predmeti();
                izabran_stub= broj +1;
                broj = brojstubova;
                Refresh();
            }
            broj++;
        }
    }
```

Igrac_vs_Igrac

Промењиве у овом делу кода су:

`Random r`-се користи да би се случајно изабрала боја за куглицу;

`Label pobeda`-док се игра информише играче ко је на потезу,а на крају игре информише играче ко је победио;

`double igrac`-одређује који је играч на потезу(1=први играч,-1=други играч);

4 int-а који одређују који стуб је кликнут,укупан број стубова на екрану,полупречник кружнице и чува број стубова на почетку игре;

2 листе-једна је листа Stub-ова, друга је листа укупног броја куглица по стубу са почетка игре.

`bool kraj-` одређује да ли се игра завршила;

```
Random r = new Random();
Label pobeda;
double igrac;
int izaban_stub, brojstubova, radijus, restart_stubovi;
List<Stub> stubovi;
bool kraj;
List<int> restart_itemi;
```

Приликом учитава овог дела програма се користи листа brojpredmeta из Pravljenje_nivoa да би се креирала листа стубова и придале им број куглица које им припадају.

```
private void Igrac_vs_Igrac_Load(object sender, EventArgs e)
        {
            this.Size = new Size(400, 400);
            radijus = (int)((0.8) * ClientRectangle.Width / 14);
            igrac = 1;
            stubovi = new List<Stub>();
            restart_itemi = new List<int>();
            List<int> lista = Pravljenje_nivoa.brojpredmeta;
            brojstubova = 0;
            while (Pravljenje_nivoa.brojpredmeta.Count() > brojstubova)
            {
                stubovi.Add(new Stub(brojstubova * 2 * radijus + 3 * radijus / 2,lista[brojstubova], radijus,
ClientRectangle.Height));
                stubovi[brojstubova].NapraviPredmete(r);
                restart_itemi.Add(lista[brojstubova]);
                brojstubova++;
            }
            brojstubova = lista.Count;
            restart_stubovi = lista.Count;
            //label pocetak
            pobeda = new Label();
            pobeda.Text = "Igra igrac:1 ";
            pobeda.Location = new Point(4 * ClientRectangle.Width / 5, 4 * ClientRectangle.Height / 9);
            pobeda.Size = new Size(ClientRectangle.Width / 5, ClientRectangle.Height / 9);
            pobeda.Font = new Font(Font.FontFamily, 12, FontStyle.Bold);
            this.Controls.Add(pobeda);
            //label kraj
            izaban_stub = 0;
        }
```

Paint je мање више сличан,али има додатак где се мења локација бројача и локација и величина label pobeda када се величина прозора промени.

```
private void Igrac_vs_Igrac_Paint(object sender, PaintEventArgs e)
        {
            radijus = (int)((0.8) * ClientRectangle.Width / 14);
            Provera();
            for (int i = 0; i < brojstubova; i++)
            {
                stubovi[i].update(e.Graphics, i * 2 * radijus + 3 * radijus / 2, radijus,
ClientRectangle.Height);

            }
            this.MinimumSize = new Size(400, 400);
            radijus = (int)((0.8) * ClientRectangle.Width / 14);
            Dugmeupdate(back, new Point(4 * ClientRectangle.Width / 5, 3 * ClientRectangle.Height / 9), new
Size(ClientRectangle.Width / 5, ClientRectangle.Height / 10));
            Dugmeupdate(delete, delete.Location = new Point(4 * ClientRectangle.Width / 5,
ClientRectangle.Height / 9), delete.Size = new Size(ClientRectangle.Width / 5, ClientRectangle.Height / 10));
            Dugmeupdate(restart, restart.Location = new Point(4 * ClientRectangle.Width / 5, 2 *
ClientRectangle.Height / 9), restart.Size = new Size(ClientRectangle.Width / 5, ClientRectangle.Height / 10));
            Brojac_update(izboritema, new Point(4 * ClientRectangle.Width / 5, 0), ClientRectangle.Width / 5);
            if (kraj)
            {
                pobeda.Location = new Point(ClientRectangle.Width / 4, 4 * ClientRectangle.Height / 10);
                pobeda.Size = new Size(11 * ClientRectangle.Width / 20, ClientRectangle.Height / 10);
                pobeda.Font = new Font(pobeda.Font.FontFamily, (pobeda.Height / 4 + 3 * pobeda.Width / 56) /
2, FontStyle.Bold);
            }
            else
            {
                pobeda.Location = new Point(4 * ClientRectangle.Width / 5, 4 * ClientRectangle.Height / 9);
                pobeda.Size = new Size(ClientRectangle.Width / 5, ClientRectangle.Height / 9);
                pobeda.Font = new Font(Font.FontFamily, 12, FontStyle.Bold);
            }
        }


    private void Brojac_update(NumericUpDown brojac, Point point, int width)
        {
            brojac.Location = point;
            brojac.Width = width;
        }

     private void Dugmeupdate(Button dugme, Point point, Size size)
        {
            dugme.Location = point;
            dugme.Size = size;
```

```
            dugme.Font = new Font(Font.FontFamily, (ClientRectangle.Height / 40 + ClientRectangle.Width / 40)
/ 2);
        }
```

Метода Provera ,за разлкику од прошле, садржи додатак for-циклус који тражи стуб са највише бројем куглица и користи тај стуб да мења вредности полупречника куглица ако прелазе границе. private void Provera()

```
    {
      int najvise_kuglica = 0;
      for (int i = 0; i < brojstubova; i++)
      {
        if (najvise_kuglica < stubovi[i].BrojPredmeta())
          najvise_kuglica = stubovi[i].BrojPredmeta();
      }
      if (0.8 * ClientRectangle.Height <= (najvise_kuglica) * 2 * radijus)
      {
        radijus = (int)((0.8) * ClientRectangle.Height / (najvise_kuglica) / 2);
      }
      if (0.8 * ClientRectangle.Width <= brojstubova * 2 * radijus)
      {
        radijus = (int)((0.8) * ClientRectangle.Width / (brojstubova) / 2);
      }
    }
```

2 догађаја MouseUp, MouseWheel раде слично што су радили у прошлом делу,али овде MouseUp још мења максимум бројача у зависности са ког је стуба отпуштен миш, а MouseWheel смањује или повећава вредност бројача у зависноти како се окреће куглица.

```
        private void Igrac_vs_Igrac_MouseUp(object sender, MouseEventArgs e)
        {
            int broj = 0;
            izaban_stub = 0;
            for (int i = 0; i < brojstubova; i++)
            {
                stubovi[i].Normalni_Predmeti();
            }
            Refresh();
            while (broj < brojstubova)
            {
                if (e.X < stubovi[broj].X + radijus / 2)
                {
                    stubovi[broj].Svetliji_Predmeti();
                    izaban_stub = broj + 1;
                    izboritema.Maximum = stubovi[broj].BrojPredmeta();
                    izboritema.Minimum = 1;
                    broj = brojstubova;
                    Refresh();
                }
                broj++;
            }
        }
        private void Igrac_vs_Igrac_MouseWheel(object sender, MouseEventArgs e)
        {
            if (e.Delta > 0 && izboritema.Value < izboritema.Maximum)
            {
                izboritema.Value++;
            }
            else if (e.Delta < 0 && izboritema.Value >= 2) izboritema.Value--;
        }
```

Igrac_vs_Igrac садржи 3 дугмета.
Дугме Obrisi(delete) брише број куглица које су се изабрале бирачем са стуба који је изабран.Ако се обрише последња куглица стуба,празан стуб ће се обрисати и избацити из листе stubovi и смањиће се brojstubova. Такође мења label pobeda да означи да је следећи играч на потезу. У случају да нема више куглица ово дугме ће нестати и label ће означити победника.

```
  private void delete_Click(object sender, EventArgs e)
        {
            Provera();
            int izbor = (int)izboritema.Value;
            if (izaban_stub - 1 == -1)
            {

                MessageBox.Show("Nisi izabrao stub");

            }
            else
            {
                izboritema.Maximum = stubovi[izaban_stub - 1].BrojPredmeta() - izbor;
                stubovi[izaban_stub - 1].Smanji(izbor);

                if (stubovi[izaban_stub - 1].BrojPredmeta() == 0)
                {
                    stubovi.RemoveAt(izaban_stub - 1);
                    brojstubova--;
                    izaban_stub = 0;
                }
                Refresh();
                igrac = -igrac;

            }
            if (!stubovi.Any())
            {
                kraj = true;
                igrac = -igrac;
                delete.Hide();
                pobeda.Text = "Pobedio je igrac:" + (1.5 - igrac / 2);
                pobeda.ForeColor = Color.BlueViolet;
                pobeda.Location = new Point(ClientRectangle.Width / 4, 4 * ClientRectangle.Height / 10);
                pobeda.Size = new Size(11 * ClientRectangle.Width / 20, ClientRectangle.Height / 10);
                pobeda.Font = new Font(Font.FontFamily, (pobeda.Height / 4 + 3 * pobeda.Width / 56) / 2,
FontStyle.Bold);
            }
            else
            {
                pobeda.Text = "Igra igrac: " + (1.5 - igrac / 2);
            }
        }
```

Pocni opet(restart) почиње игру испочетка и у случају да се игра завршила враћа дугме delete.

```
private void restart_Click(object sender, EventArgs e)
        {
            izaban_stub = 0;
            pobeda.Location = new Point(4 * ClientRectangle.Width / 5, 4 * ClientRectangle.Height / 9);
            pobeda.Size = new Size(ClientRectangle.Width / 5, ClientRectangle.Height / 9);
            pobeda.Font = new Font(Font.FontFamily, 12, FontStyle.Bold);
            pobeda.ForeColor = Color.Black;
            pobeda.Text = "Igra igrac:1 ";
            igrac = 1;
            if (kraj)
            {
                kraj = false;
                delete.Show();
            }
            stubovi = new List<Stub>();
            for (int i = 0; i < restart_stubovi; i++)
            {
                stubovi.Add(new Stub(i * 2 * radijus + 3 * radijus / 2, restart_itemi[i], radijus,
ClientRectangle.Height));
                stubovi[i].NapraviPredmete(r);
            }
            brojstubova = restart_stubovi;
            Refresh();
        }
```

Дугме Nazad(back) враћа корисника на `Pravljenje_nivoa` и затвара овај програм.

```
private void back_Click(object sender, EventArgs e)
    {
      this.Hide();
      Pravljenje_nivoa napravi_nivo = new Pravljenje_nivoa();
      napravi_nivo.ShowDialog();
      this.Close();
 }
```

Igrac_vs_kompjuter

Овај програм има доста сличности са Igrac_vs_igrac .
Уз променљиве ис прошлог дела има и још:
`bool radnja`-провера да ли компјутер прави свој потез и ако прави блокираће корсника да притисне на било које дугме док компјутер не заврши свој потез.
`Label ai_potez`-садржи текст који говори играчи који потез да направи да не би предао вођство компјутеру, ако је већ предато вођство обавестиће играча.
`int broj_pomoci`-чува број колико пута може да се притисне дугме Pomoc док не нестане.
Методе су мање више исте. При покретању програма се исто учитава као Igrac_vs_igrac. Paint додатно мења и локацију и величину нове label.
Delete дугме сада извршава потез и играча и компјутера. Async је додато како би могла да се користи заустављање кода на 1.5 секунде да би играч могао да види какав је терен направио пре него што га компјутер такође прмени. Кад се игра заврши, позива се метода Крај која избацује label pobeda са текстом ко је победиом и сакрива дугме delete и Pomoc(help).

```
async private void delete_Click(object sender, EventArgs e)
{
    this.Controls.Remove(ai_potez);
    Provera();
    int izbor = (int)izboritema.Value;
    if (radnja == true)
    {
        MessageBox.Show("Sacekaj da kompjuter napravi svoj potez");
    }
    else if (izabran_stub == 0)
    {
        MessageBox.Show("Nisi izabrao stub");
    }

    else
    {
        radnja = true;
        izboritema.Maximum = stubovi[izabran_stub - 1].BrojPredmeta() -izbor;
        stubovi[izabran_stub - 1].Smanji(izbor);

        if (stubovi[izabran_stub - 1].BrojPredmeta() == 0)
        {
            stubovi.Remove(stubovi[izabran_stub - 1]);
            brojstubova--;
        }
        izabran_stub = 0;
        izabran_stub = 0;
        if (brojstubova == 0)
            radnja = false;
        Refresh();
        if (!stubovi.Any())de
        {
            Kraj(true);
        }
        if (radnja)
        {
            List<int> lista_itema = new List<int>();
            int broj_kuglica = 0;
            int dubina_max = 0;
            for (int i = 0; i < brojstubova; i++)
            {
                lista_itema.Add(stubovi[i].BrojPredmeta());
                broj_kuglica += stubovi[i].BrojPredmeta();
            }
            if (broj_kuglica <= 15)
                dubina_max = 5;
            else if (broj_kuglica < 30 && broj_kuglica > 15)
                dubina_max = 3;
            else dubina_max = 2;
            Min_max algoritam = new Min_max(dubina_max);
            int vredonst=algoritam.Minmax(true, lista_itema, 0, -500, 500);
            int ai_stub = algoritam.broj_stuba;
            int ai_kuglice = algoritam.broj_kuglica;
            await Task.Delay(1500);
            izboritema.Maximum = stubovi[ai_stub].BrojPredmeta() - ai_kuglice;
```

```
                stubovi[ai_stub].Smanji(ai_kuglice);
                if (stubovi[ai_stub].BrojPredmeta() == 0)
                {
                    stubovi.Remove(stubovi[ai_stub]);
                    brojstubova--;

                }
                for (int i = 0; i < brojstubova; i++)
                {
                    stubovi[i].Normalni_Predmeti();
                }

                radnja = false;
                Refresh();
                if (!stubovi.Any())
                {
                    Kraj(false);

                }
            }
        }
    }
    private void Kraj(bool igrac)
    {
        kraj = true;
        delete.Hide();
        hint.Hide();
        pobeda = new Label();
        if (igrac)
        {
            pobeda.ForeColor = Color.BlueViolet;
            pobeda.Text = "Pobedio je igrac ";
        }
        else
        {
            pobeda.Text = "Izgubio si ";
            pobeda.ForeColor = Color.Red;
        }
        pobeda.Location = new Point(ClientRectangle.Width / 4, 4 * ClientRectangle.Height / 10);
        pobeda.Size = new Size(11 * ClientRectangle.Width / 20, ClientRectangle.Height / 10);
        pobeda.Font = new Font(Font.FontFamily, (pobeda.Height / 4 + 3 * pobeda.Width / 56) / 2, FontStyle.Bold);
        this.Controls.Add(pobeda);
    }
}
```

Овај програм садржи додатно дугме help. То дугме враћа label ai_potez која садржи текст који каже какав потез треба да се направи да се не изгуби вођство. Ако је вођство већ игубљено онда ће се вратити текст”XXXXXX”. После 3 клика ће дугме нестати.

```
    private void pomoc_Click(object sender, EventArgs e)
    {
        if (radnja)
            MessageBox.Show("Sacekaj da kompjuter zavrsi svoj potez");
        else if (broj_pomoci == 0)
        {
            hint.Hide();
        }
        else
        {
            broj_pomoci--;
            List<int> lista_itema = new List<int>();
            int broj_kuglica = 0;
            int dubina_max = 0;
            for (int i = 0; i < brojstubova; i++)
            {
                lista_itema.Add(stubovi[i].BrojPredmeta());
                broj_kuglica += stubovi[i].BrojPredmeta();
            }
            if (broj_kuglica <= 15)
                dubina_max = 5;
            else if (broj_kuglica < 30 && broj_kuglica > 15)
                dubina_max = 3;
            else dubina_max = 2;
            Min_max algoritam = new Min_max(dubina_max);
            int vrednost = algoritam.Minmax(true, lista_itema, 0, -500, 500);
            if (vrednost < 0)
            {
                this.Controls.Remove(ai_potez);
                ai_potez = new Label();
                ai_potez.Text = "XXXXXX";
                ai_potez.Location = new Point(4 * ClientRectangle.Width / 5, 5 * ClientRectangle.Height / 9);
                ai_potez.Size = new Size(ClientRectangle.Width / 5, ClientRectangle.Height / 10);
```

```
                ai_potez.Font = new Font(Font.FontFamily, 11, FontStyle.Bold);
                this.Controls.Add(ai_potez);
            }
            else
            {
                int ai_stub =algoritam.broj_stuba;
                int ai_kuglice = algoritam.broj_kuglica;
                if (ai_potez != null)
                    this.Controls.Remove(ai_potez);
                ai_potez = new Label();
                ai_potez.Text = "stub:" + (ai_stub + 1) + "\n" + "kuglica:" + ai_kuglice;
                ai_potez.Location = new Point(4 * ClientRectangle.Width / 5, 5 * ClientRectangle.Height / 9);
                ai_potez.Size = new Size(ClientRectangle.Width / 5, ClientRectangle.Height / 10);
                ai_potez.Font = new Font(Font.FontFamily, 11, FontStyle.Bold);
                this.Controls.Add(ai_potez);
            }
        }
    }
```

Back и restart дугме раде исто што су радили и у Igrac_vs_igrac са додатком да restart враћа сада и дугме help ако с игра завршила.

Садржај

Литература

1.Winning Ways For Your Mathematical Plays Volume 1, Elwyn R. Berlekamp, John H. Conway, and Richard K. Guy
2.Tic Tac Toe:Understanding The Minimax Algorithm, http://neverstopbuilding.com/minimax