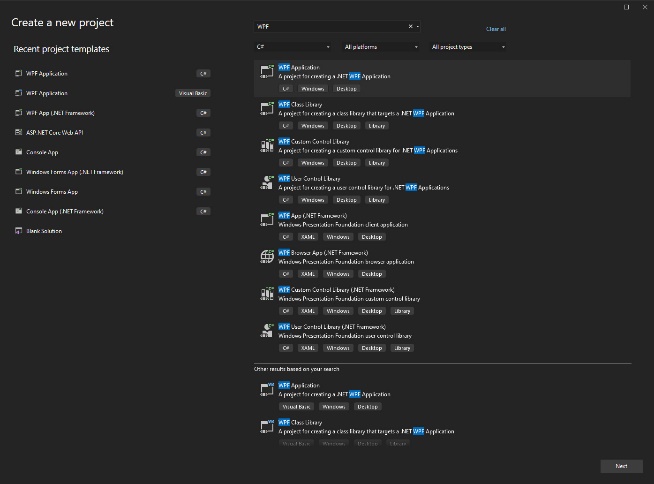
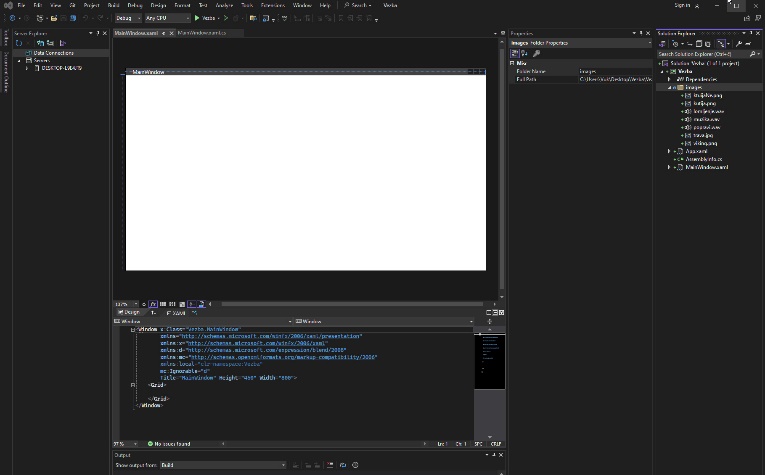
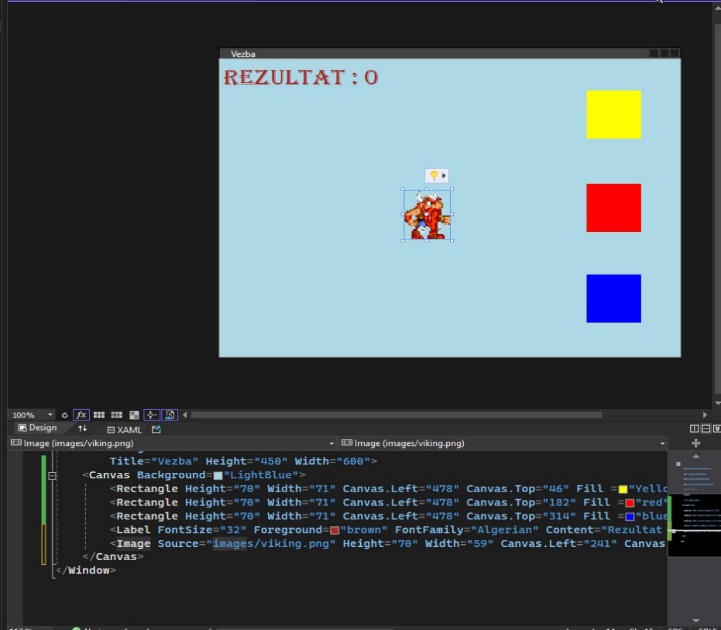
Vežba

# Namestanje slike

Otvorimo WPF Projekat i napravimo folder za slike i u njega ubacimo sve slike. OBAVEZNO PODESITI SLIKA NA RESOURCE U PROPERTY-U ,posto moze da se desi da se ne prikazuju slike !

Radicemo sve u MainWindow.xaml i MainWindow.xaml.cs, App.xaml nam vise sluzi za opstu stilizaciju... MainWindow.xaml moze da gledamo kao na HTML dok na App.XAML gledamo kao CSS

U WPF-u na glavnoj strani za dizajn vidimo slikovnu reprezentaciju koda... kao i kod u vidu XAML fajla, XAML je veoma sličan HTML, imamo određene atribute unutar tagova... Za radnju sa WPF-om, nama su najznačajni: **Rectangle, Canvas, Image i Label. Canvas je kontejner koji nam sluzi za apsolutno pozicioniranje elemenata.** sve sto radimo u WPF-u obuhvatamo Canvasom, **Grid** nam sluzi vise za pravilnije oblike za postavljanje elemenata u kolone i redove, npr. pravimo sudoku iks/oks... Atributi Canvas.Left, Canvas.Right, Canvas.top, na pokzauju koliko je element udaljen od od dela Canvasa opisan u atributu, znaci Canvas.top koliko je udaljen od gornje ivice Canvasa. Atributom Fill bojimo pozadinu elementa. Label nam sluzi za tekst koji ne mozemo da menjamo, odnosno u formi kad aktiviramo aplikaciju ne mozemo da menjamo, mozemo da menjamo u .CS, takodje tekst je potrebno napisati u ***Content*** atributu ne unutar taga

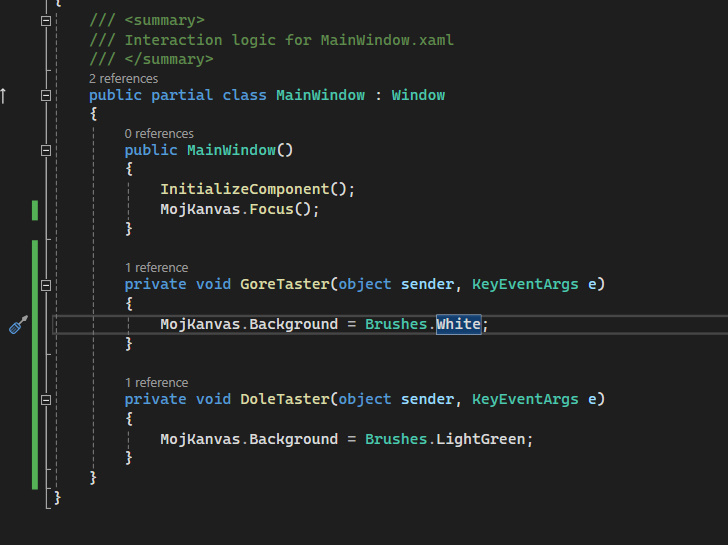
# Name/Tag

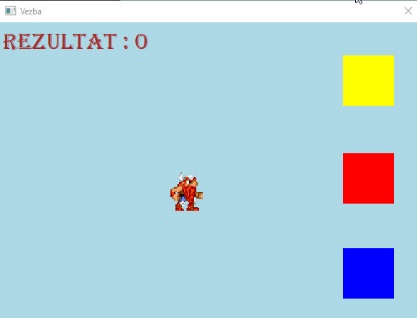
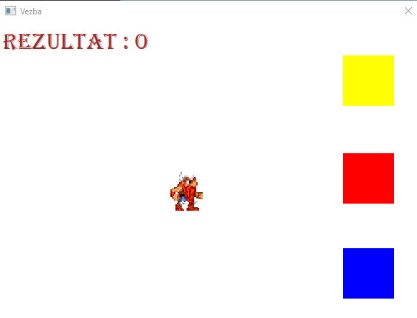
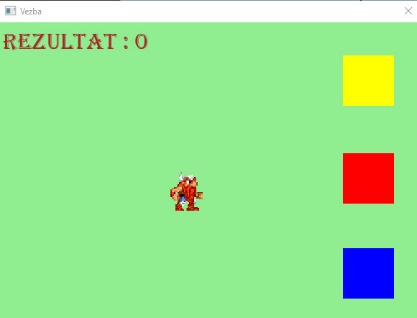
**Name** nam sluzi za identifikaciju specificnog elementa (Ime kod svakog mora da bude jedinstveno), sa druge strane **Tag** nam da obuhvatimo vise razlicitih elemenata, koji zelimo da se nesto slicno desi. A screen shot of a computer program

Description automatically generated

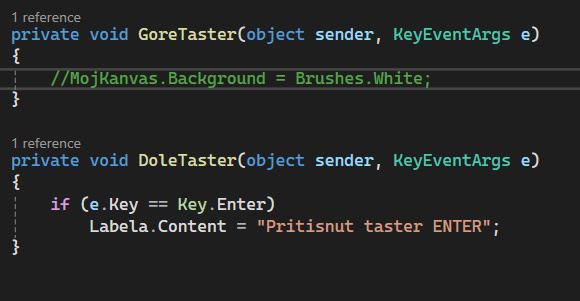
# 3.Koriscenje tastera

Postoje dva dogadjaja vezana za tastere KeyUp i keyDown, KeyUp je dogadjaj Pustanja tastera, dok je keyDown spustanje tastera. NEOPHODNO je da u tag Canvasa stavimo Focusable = true i MojKanvas.Focus u konstruktoru, da bismo mogli da omogucimo da aplikacija primeti da se desio dogadjaj vezan za taster. Metode vezane za dogadjaj se izvrsavaju ne bitno koji taster je u pitanju

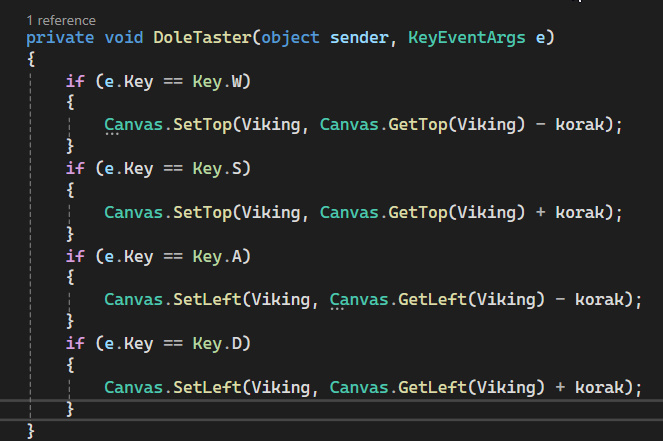




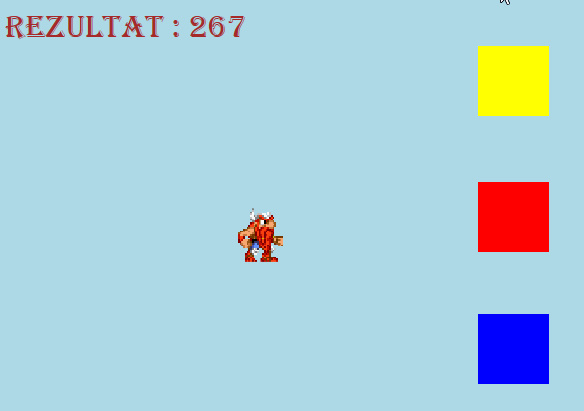
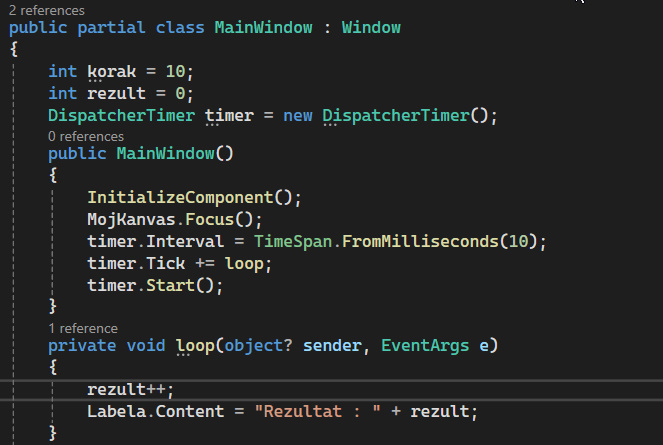
Da bismo obezbedili da se izvrsi metoda za specificni taster prosto treba da koristimo IF



# 4.Kretanje Vikinga

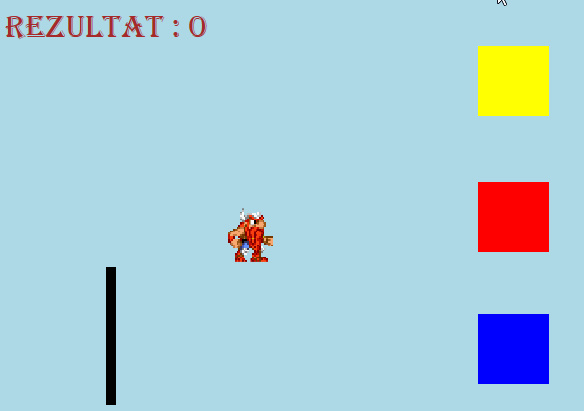
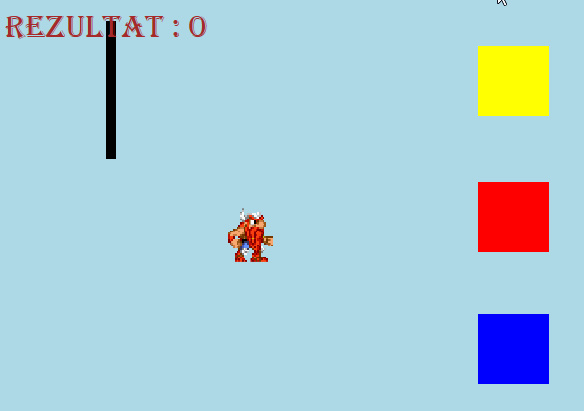
Da bi smo napravili kreirali kretanje elementa viking, moramo da koristimo kanvas, Canvas.SetLeft() ova metoda prihvata 2 argumenta, element kojem namestamo Canvas udaljenost i novu vrednost, Za ovo drugo koristimo: Canvas.GetLeft() - ovo samo vraca udaljenost od leve ivice canvasa

# 5.DispatcherTimer

Timer u WPF-u je malo komplikovaniji nego u WinFormi, mora da se pravi unutar .cs fajla... Mora da se namesti interval izmedju 2 tika, i da se doda metoda na tik dogadja, i obavezno se pokrene tajmer

## ZADATAK 1.

Koristeci znanje iz winForme i ovo sto ste sad naucili, napraviti Element imena „Granica“ tipa Rectangle i napraviti da se on stalno pomera gore/dole, Ukoliko udari donju/gornju granicu promeni mu se smer kretanja.



## RESENJE

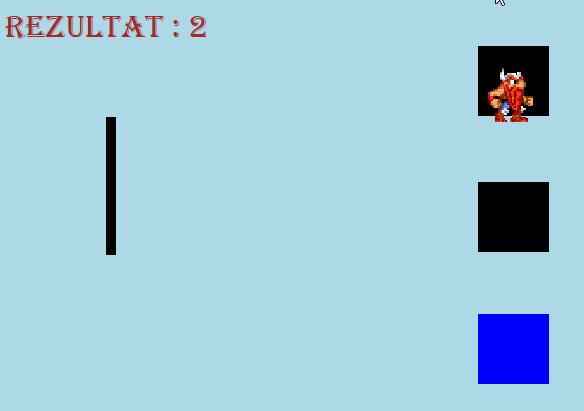
# 6. Povezivanje timera sa KeyUp i KeyDown dogadjajem

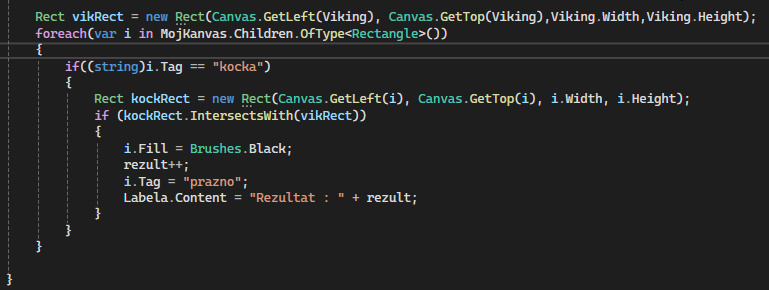
Kao sto vidimo kretanje nije bas idealno sa KeyDown dogadjajem, sporo je, i ne moze ukoso da pokrecemo lika. Da bi smo ovo poboljsali moramo da iskoristimo timer, i keyUp. KeyDown metodom cemo napraviti da se bool promenljiva za odredjeni smer, preko tajmera cemo napraviti da se element pokrene u odredjenom smeru pri svakom tiku, dok je bool postavljen na true, dok ce se bool vratiti na false, ako se pusti taster.

A screenshot of a computer program

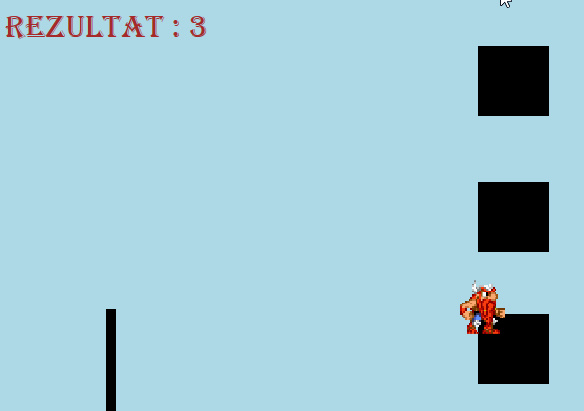
Description automatically generated

# 7. Rect Struktura

Na rect Strukturu mozemo da gledamo kao na pravougaonik koji se stvara ne mesto elementa koji se stavi u konstruktor (odnosno: Rect r = new Rect(Canvas.GetLeft(Viking), Canvas.GetTop(Viking),Viking.Width,Viking.Height). Ono sto cini rect dosta korisnim je metoda .IntersectsWith (pokazuje da li se dva elementa seku) kao i .Contains (pokazuje da li se jedan element nalazi u drugom), ovo moze da nam pokaze da li se element sudara sa drugim elementom

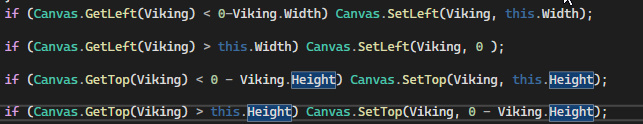


## ZADATAK 2.

Napraviti da kad viking izadje iz granica forme, se ponovo stvori da suprotnoj straniA screenshot of a video game

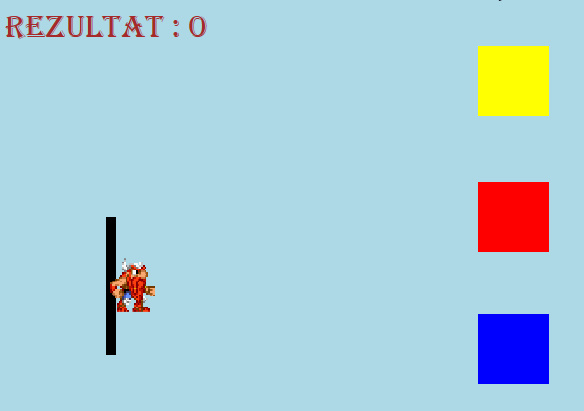
Description automatically generated

## RESENJE

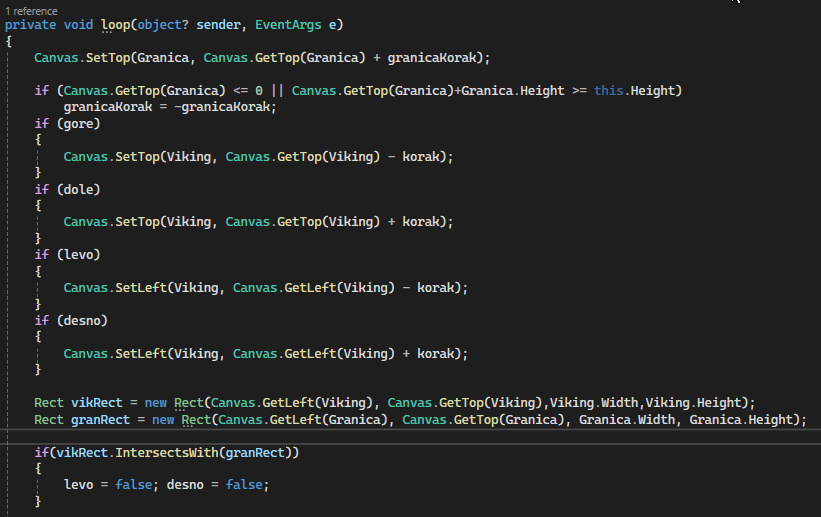
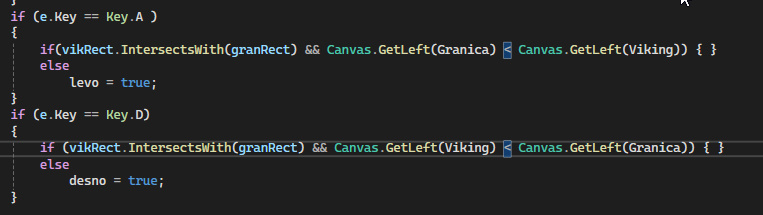


## ZADATAK 3.

Napraviti da Viking ne moze da prodje kroz element Granica (Samo levo/desno)



## RESENJE



## DOMACI

Napraviti da viking moze da se krece levo desno samo kad je na platformi (Gornja/donja) i da se krece gore dole kad je na merdevinama

