Zadaća 3

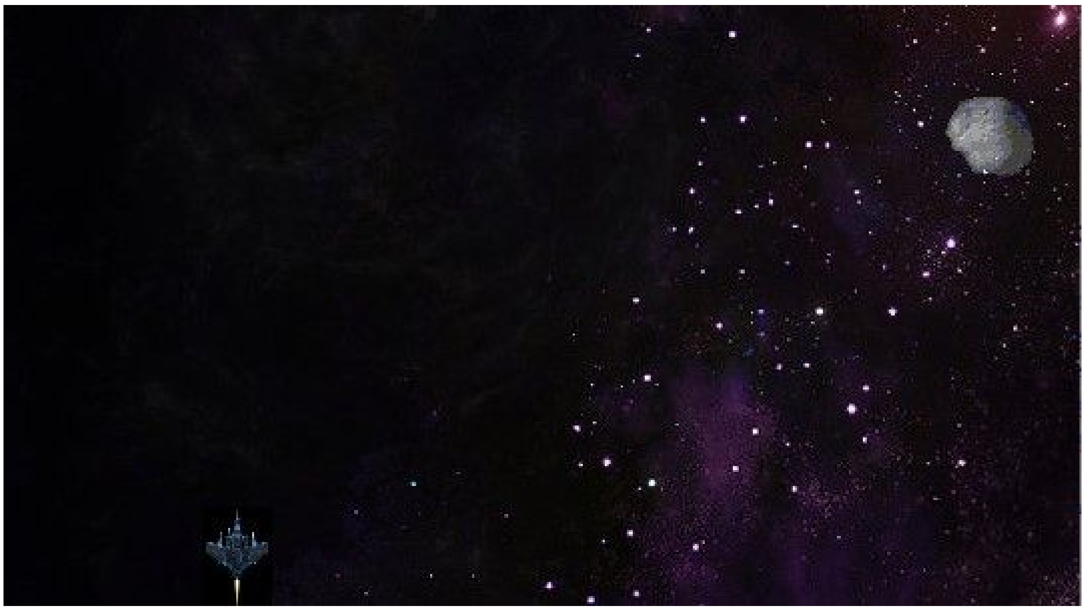
Zadatak 1

Napisati program koji će animirati kretanje svemirskog broda u svemiru. Proizvoljno uzeti ili nacrtati svemirski brod. Brod će se kretati prema gore, proizvoljnom brzinom. Početna pozicija broda je u donjem dijelu prozora. Kada brod u potpunosti izađe iz okvira prozora sa gornje strane, brod je potrebno ponovo postaviti u početnu poziciju, te nastaviti postupak.

Osim toga, svakih **10 sekundi**, novi asteroid će se kreirati i početi svoje kretanje. Asteroid će se kretati prema dole, proizvoljnom brzinom i u jednom trenutku će izaći van okvira prozora. proizvoljno uzeti ili nacrtati asteroid, te simulirati njegovo kretanje.

Napraviti nekoliko slika za animaciju broda i asteroida (npr. kod asteroida je moguće kreirati slike sa sitnom razlikom u izgledu asteroida da bi izgledalo kao da se asteroid okreće, dok je kod broda moguće dodati animaciju dima). Obratiti pažnju na transparentnost prilikom crtanja pojedinih objekata.

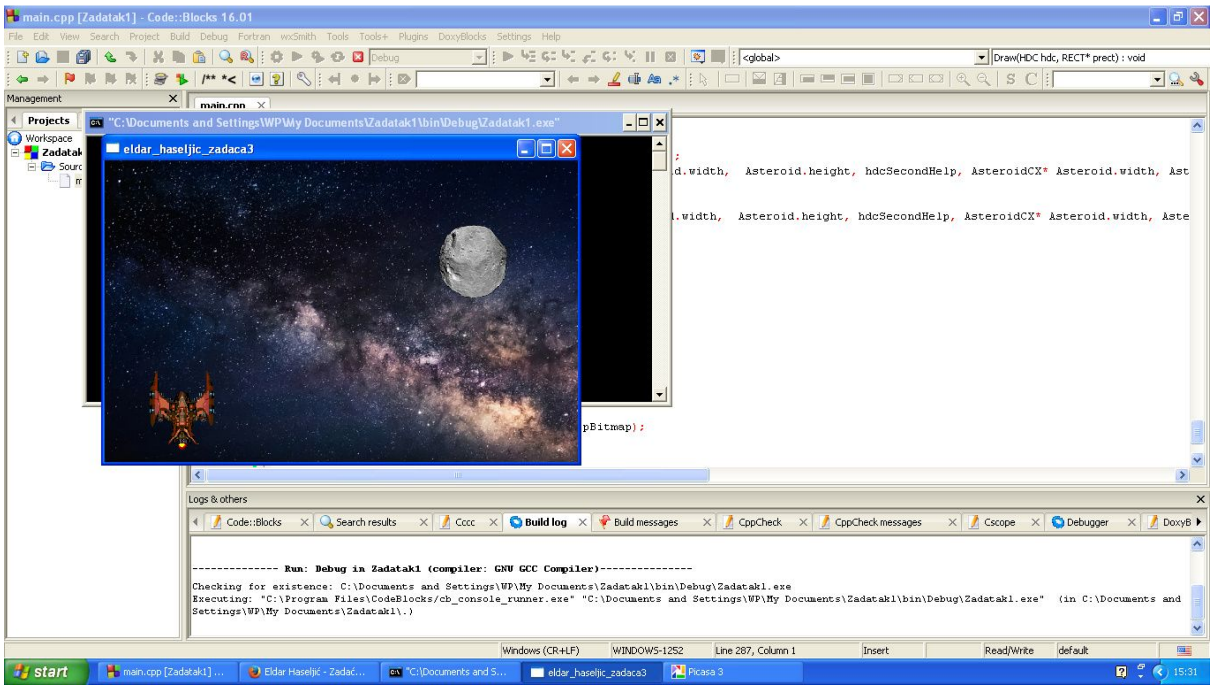
Mogući izgled aplikacije:



Nakon završetka zadatka, upload – ovati .zip fajl sa CodeBlocks projektom i svim resursima (slikama) koji su korišteni u projektu pri čemu zip file imenovati na sljedeći način:

ime_prezime_zadaca3.zip

Slika prozora (screen shot)



```
d.width,  Asteroid.height, hdcSecondHelp, AsteroidCX* Asteroid.width, Ast

.width,  Asteroid.height, hdcSecondHelp, AsteroidCX* Asteroid.width, Aste

pBitmap);
```

```
-------------- Run: Debug in Zadatak1 (compiler: GNU GCC Compiler)---------------
Checking for existence: C:\Documents and Settings\WP\My Documents\Zadatak1\bin\Debug\Zadatak1.exe
Executing: "C:\Program Files\CodeBlocks/cb_console_runner.exe" "C:\Documents and Settings\WP\My Documents\Zadatak1\bin\Debug\Zadatak1.exe"  (in C:\Documents and
Settings\WP\My Documents\Zadatak1\.)
```

Definicija window klase

```
 HWND hwnd;               /* This is the handle for our window */
      MSG messages;            /* Here messages to the application are saved
*/
      WNDCLASSEX wincl;        /* Data structure for the windowclass */

      /* The Window structure */
      wincl.hInstance = hThisInstance;
      wincl.lpszClassName = szClassName;
      wincl.lpfnWndProc = WindowProcedure;      /* This function is called
by windows */
      wincl.style = CS_VREDRAW | CS_HREDRAW;                 /* Catch
double-clicks */
      wincl.cbSize = sizeof (WNDCLASSEX);

      /* Use default icon and mouse-pointer */
      wincl.hIcon = LoadIcon (NULL, IDI_APPLICATION);
      wincl.hIconSm = LoadIcon (NULL, IDI_APPLICATION);
      wincl.hCursor = LoadCursor (NULL, IDC_ARROW);
      wincl.lpszMenuName = NULL;                /* No menu */
      wincl.cbClsExtra = 0;                     /* No extra bytes after the
window class */
      wincl.cbWndExtra = 0;                     /* structure or the window
instance */
```

```
      /* Use Windows's default colour as the background of the window */
      wincl.hbrBackground = (HBRUSH) COLOR_BACKGROUND;

      /* Register the window class, and if it fails quit the program */
      if (!RegisterClassEx (&wincl))
      return 0;
```

Poziv funkcije CreateWindow

```
 /* The class is registered, let's create the program*/
      hwnd = CreateWindowEx (
            0,               /* Extended possibilites for variation */
            szClassName,     /* Classname */
            _T("eldar_haseljic_zadaca3"),        /* Title Text */
            WS_OVERLAPPEDWINDOW, /* default window */
            CW_USEDEFAULT,     /* Windows decides the position */
            CW_USEDEFAULT,     /* where the window ends up on the screen */
            544,               /* The programs width */
            375,               /* and height in pixels */
            HWND_DESKTOP,      /* The window is a child-window to desktop */
            NULL,              /* No menu */
            hThisInstance,     /* Program Instance handler */
            NULL               /* No Window Creation data */
            );
```

Definicija window procedure

```
/*  This function is called by the Windows function DispatchMessage()  */

LRESULT CALLBACK WindowProcedure (HWND hwnd, UINT message, WPARAM wParam,
LPARAM lParam)
{

      switch (message)               /* handle the messages */
      {
      case WM_SIZE:
      {
      BITMAP bitmap;
      SetTimer(hwnd, ID_TIMER1, 10, NULL);
      hbmSky =
(HBITMAP)LoadImage(NULL,"sky.bmp",IMAGE_BITMAP,LOWORD(lParam),HIWORD(lParam),
LR_LOADFROMFILE);
      hbmSpaceship =
(HBITMAP)LoadImage(NULL,"spaceship_black.bmp",IMAGE_BITMAP,0,0,LR_LOADFROMFIL
E);
```

```
      hbmSpaceshipMask =
(HBITMAP)LoadImage(NULL,"spaceship_white.bmp",IMAGE_BITMAP,0,0,LR_LOADFROMFIL
E);

      SetTimer(hwnd,ID_TIMER2, 10000, NULL);
      hbmAsteroid =
(HBITMAP)LoadImage(NULL,"asteroid_black.bmp",IMAGE_BITMAP,0,0,LR_LOADFROMFILE
);
      hbmAsteroidMask =
(HBITMAP)LoadImage(NULL,"asteroid_white.bmp",IMAGE_BITMAP,0,0,LR_LOADFROMFILE
);

      GetObject(hbmSpaceship,sizeof(BITMAP),&bitmap);

      Spaceship.width = bitmap.bmWidth/3;
      Spaceship.height = bitmap.bmHeight;
      Spaceship.dx = (LOWORD(lParam)-bitmap.bmWidth/3)/2;
      Spaceship.y = HIWORD(lParam)-bitmap.bmHeight;
      Spaceship.dy = 1;

      GetObject(hbmAsteroid,sizeof(bitmap),&bitmap);
      Asteroid.width = bitmap.bmWidth/2;
      Asteroid.height = bitmap.bmHeight/2;
      Asteroid.x = (LOWORD(lParam)-bitmap.bmWidth/2);
      Asteroid.y = 0;

      GetObject(hbmSky,sizeof(bitmap),&bitmap);

      Sky.width = bitmap.bmWidth;
      Sky.height = bitmap.bmHeight;
      Sky.dx = 0;
      Sky.dy = 0;

      break;
      }
      case WM_TIMER:
      {
      RECT clientRectangle;
      HDC hdc;
      switch(wParam)
      {
      case ID_TIMER1:
      {
            hdc = GetDC(hwnd);
            GetClientRect(hwnd, &clientRectangle);

            MoveSpaceship(&clientRectangle);
            MoveAsteroid(&clientRectangle);
            Draw(hdc, &clientRectangle);

            ReleaseDC(hwnd, hdc);
```

```
        ++i;
        if(i == 5)
        i =0;
        break;
    }
    case ID_TIMER2:
    {
        hdc = GetDC(hwnd);
        GetClientRect(hwnd, &clientRectangle);
        Asteroid.x = clientRectangle.right-Asteroid.width ;
        Asteroid.y = clientRectangle.top ;
        ReleaseDC(hwnd, hdc);
        break;
    }
    }
    break;
    }
    case WM_KEYDOWN:
    {
    switch(wParam)
    {
    case VK_DOWN:
        if(Spaceship.dy > 1)
        --Spaceship.dy;
        break;
    case VK_UP:
        ++Spaceship.dy;
        break;
    case VK_LEFT:
        if(Spaceship.x>0)
        Spaceship.dx -= 15;
        break;
    case VK_RIGHT:
        RECT rect;
        GetClientRect(hwnd,&rect);
        if(Spaceship.x < rect.right - Spaceship.width)
        Spaceship.dx += 15;
        break;

    }
    break;
    }
    case WM_DESTROY:
    KillTimer(hwnd,ID_TIMER1);
    KillTimer(hwnd,ID_TIMER2);
    PostQuitMessage (0);     /* send a WM_QUIT to the message queue */
    break;
    default:                       /* for messages that we don't deal with
*/
    return DefWindowProc (hwnd, message, wParam, lParam);
```

```
        }

        return 0;
}
```