A fény viselkedésének számítógépes modellezése

Jelige: goifd2018

Egy olyan alkalmazásnak az ötlete, ami lehetővé tenné, hogy egy fénysugár útját kövessük, miközben különféle optikai berendezésekkel lép kapcsolatba, körülbelül egy évvel ezelőtt merült föl egy fizikaszakkör alkalmával. Ezt az ötletet dolgoztam most ki.

Elmélet

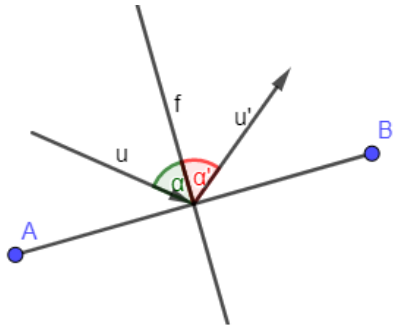
I. Fizikai elemek

Az alkalmazáson belül három eszközt lehet használni: síktükröt, gömbtükröt és lencsét.

Síktükör *(1.ábra)*

Az *f* egyenes a síktükör beesési merőlegese, *u* a síktükörhöz közeledő fénysugár, *u'* pedig a visszavert fénysugár. A beesési szög *α*, a közeledő fénysugár és a beesési merőleges által bezárt szög megegyezik a visszavert fénysugár és a beesési merőleges által bezárt szöggel, ami *α'*.

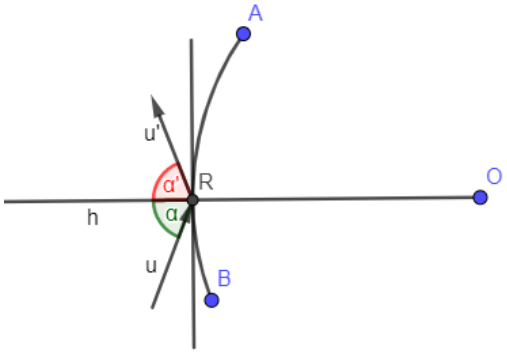
α = α'



1. ábra.
Fény visszaverődése síktükörről.

Gömbtükör *(2. ábra)*

A gömbtükör azon *R* pontjába, melybe a fénysugár érkezik, érintőt szerkeszthetünk. Ezzel az érintővel helyettesíthetjük a gömbtükür felületét, az érintő, mint síktükör veri vissza a fényt. A körhöz húzott érintő merőleges az érintési pontba húzott sugárra, így az *O* végpontú, *R* ponton áthaladó félegyenes a beesési merőleges, ahol *O* a gömbtükör középpontja. A beesési szög *α*, csakúgy, mint a korábbi esetben, megegyezik a visszaverődési szöggel, *α'*. Akár homorú akár domború a gömbtükör, az eljárás nem változik.*α = α'*

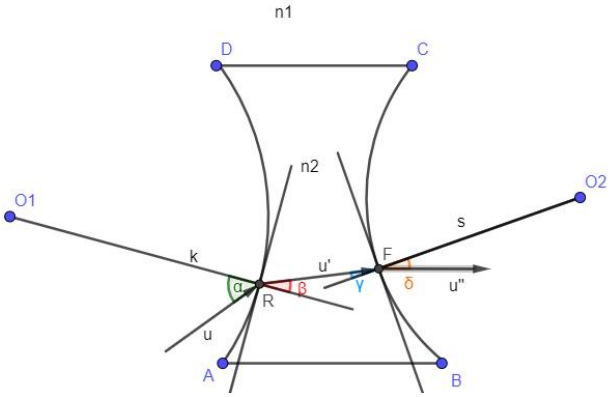


2. ábra.
Fény visszaverődése gömbtükörről.

Lencse *(3. ábra)*

Az alkalmazásban nem különítettem el a vékony és vastag lencséket, így itt sem fogom.

A Snellius-Descartes törvény értelmében a fény új közeghatárhoz érve megtörik. Az új közeghatár beesési merőlegese *k*, *k* merőleges a körív R pontjába húzott



3. ábra.
Fény törése lencsén.

érintőjére és a fény vele α szöget zár be. Törés után a fénysugár a beesési merőlegessel β szöget zár be.

α és β szög kapcsolatát az alábbi egyenlet írja le:

$\sin\alpha * n_1 = \sin\beta * n_2$. n_1 és n_2 a külső illetve belső közeg törésmutatója.

$\beta > 90^\circ$ esetén a felület, mint tükör kezelendő a fent leírtak alapján.

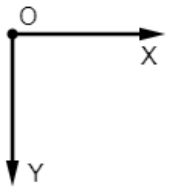
A második törés esetén illetve domború lencse esetén is ez az eljárás.

II. Matematikai megvalósítás

A számítógépes megvalósítás nagy mértékben megnehezíti az egyébként nem olyan bonyolult matemaikai részét a problémának. Magát az alkalmazást később még részletesen leírom, viszont ahhoz, hogy a matematikai részét bemutathassam, néhány részletet az alábbi pontokban ismertetek.

Koordinátarendszer *(4. ábra)*

A megjelenítéshez használt koordinátarendszer origója a megjelenítési felület bal felső sarkában található. Az *x* tengely jobbra növekszik, az *y* tengely pedig lefelé.



4. ábra.
Koordinátarendszer.

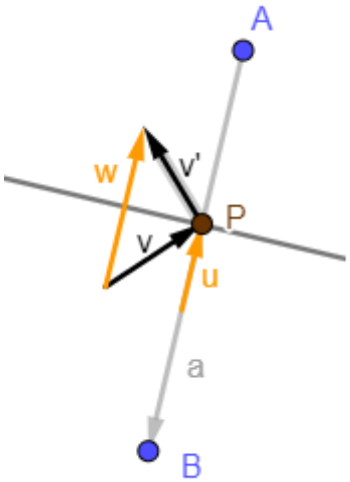
Fénysugár

A fénysugarat a programban egy kétdimenziós vektor és egy pont képviseli. A pontot t időnként eltolom a vektorral, így a pont koordinátái megváltoznak. Ha azt akarom, hogy a fénysugár irányt változtasson, akkor a vektor irányát kell megváltoztatni.

$$P\,(u;v),\ \vec{V}\begin{bmatrix}x\\y\end{bmatrix} \qquad P\,(u;v) \rightarrow P\,(u+x;v+y)$$

Síktükör *(5. ábra)*

Egy síktükröt az alkalmazáson belül két pont és az ezeket összekötő szakasz reprezentál. Amikor a fény pontjának koordinátái megegyeznek a szakasz bármely pontjának koordinátáival, akkor "ütközik" a fénysugár a tükörbe. Legyen az ütközési pont P, a tükör két végpontja pedig A illetve B, az ezeket összekötő szakasz a, $\vec{v}$ pedig a fényhez tartozó vektor az ütközés előtt. $\vec{v}$-t levetítve $\overrightarrow{AB}$-ra megkapjuk $\vec{u}$-t. $\vec{u}$-t megkétszerezve és ebből $\vec{v}$-t kivonva megkapjuk $\overrightarrow{v'}$-t.



5. ábra.
Vektorműveletek szemléltetése ábrával.

$$\text{proj}_{\overrightarrow{AB}}\,\vec{v} = \vec{u} = \left(\frac{\vec{v} * \overrightarrow{AB}}{\left|\overrightarrow{AB}\right|^2}\right) * \overrightarrow{AB}, \qquad \vec{w} = 2 * \vec{u}$$

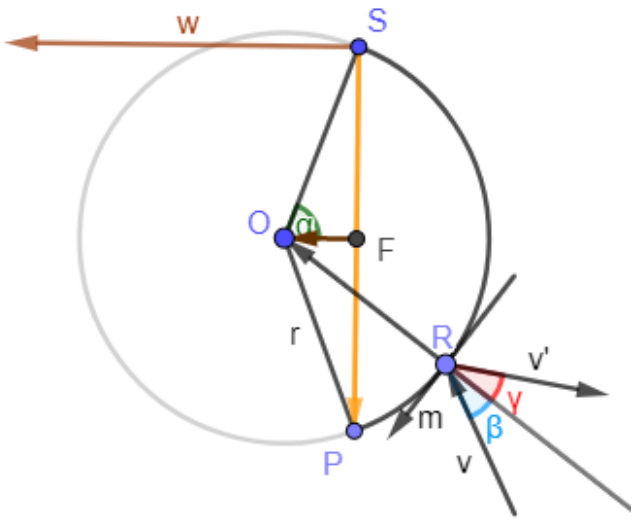
$$\vec{w} = 2 * \left(\frac{\vec{v} * \overrightarrow{AB}}{\left|\overrightarrow{AB}\right|^2}\right) * \overrightarrow{AB}\,, \qquad \overrightarrow{v'} = \vec{w} - \vec{v}$$

$$\vec{v'} = 2 * \left(\frac{\vec{v} * \overrightarrow{AB}}{|\overrightarrow{AB}|^2} \right) * \overrightarrow{AB} - \vec{v}$$

Gömbtükör *(6. ábra)*

Egy gömbtükröt az alkalmazásban két pont és egy számérték megadásával lehet létrehozni. A két pont a körív két végpontja, a számérték pedig annak a körnek a sugara, amiből a két pont kimetszi ezt az ívrészt. Habár a két pont a teljes körívet két részre osztja, a program mindig a rövidebbet azonosítja a tükörrel, ez viszont azt is jelenti, hogy a gömbtükörhöz tartozó körív mindig kisebb, mint egy félkör. A fizikai résznél már volt arról szó, hogy attól kezdve, hogy az ív azon pontjába érintőt húzunk, ahol a fénysugár "ütközik" a tükörrel, helyettesíthető a gömbtükör egy síktükörrel, így, ha ez sikerül, attól kezdve már a síktükörnél leírtak használhatóak.

Legyen a körív két végpontja S és P, sugara r, R pedig az "ütközés" pontja. Először számoljuk ki a kör középpontjának koordinátáit:



6. ábra.

$\sin \alpha = \frac{|\overrightarrow{SP}|}{2*r} \rightarrow \alpha = \sin^{-1} \frac{|\overrightarrow{SP}|}{2*r}$

$|\overrightarrow{FO}| = r * \cos \alpha \rightarrow |\overrightarrow{FO}| = r * \cos\left(\sin^{-1} \frac{|\overrightarrow{SP}|}{2*r}\right)$

$\overrightarrow{SP}$-t forgassuk el 90˚- kal negatív irányba

legyen ez a vektor $\vec{w}$.

$\vec{w} \parallel \overrightarrow{FO} \rightarrow |\vec{w}| * k = |\overrightarrow{FO}|$

ahol k az a szám amivel $\vec{w}$ – t megszorozva $\overrightarrow{FO}$ – t kapjuk.

Számoljuk ki SP szakasz felezőpontját:

$F = \frac{S+P}{2}$

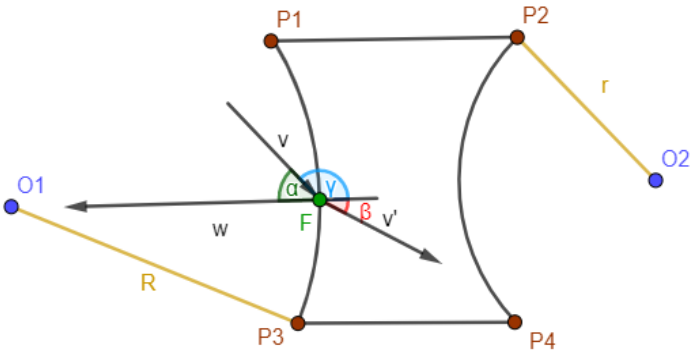
Majd számoljuk ki a kör középpontját, O-t úgy, hogy F pontot eltoljuk $\overrightarrow{FO}$-val.

$\underline{O = F + \vec{w} * k}$

$\overrightarrow{RO}$ –t 90˚- kal negatív irányba elforgatva megkapjuk $\vec{m}$-t, amit a síktükör esetében $\overrightarrow{AB}$ – vel jelöltünk. Ez az a vektor, ami a visszaverődési felület egyenesével párhuzamos. $\vec{v}$ továbbra is a beeső fénnysugár, $\vec{v'}$ peig a távozó. $\beta = \gamma$

Innentől az $\vec{v'} = 2 * \left(\frac{\vec{v} * \overrightarrow{AB}}{|\overrightarrow{AB}|^2} \right) * \overrightarrow{AB} - \vec{v}$ egyenletbe behelyettesítve megkapjuk a fénysugár új irányát.

Lencse *(7. ábra)*



7. ábra.

A lencsék (homorú és domború) a legbonyolultabb alakzatok az alkalmazásban, létrehozásukhoz 6 adat szükséges, illetve még eldönthető, hogy homorú vagy domború lencsét akar-e a felhasználó létrehozni. Az adatok a következők: két görbületi sugár, valamint a lencse négy csúcspontja. Az ábrán r, R és P_1, P_2, P_3, P_4, a külső közeg és a lencse törésmutatóját indításkor meg kell adni. A gömbtükörnél leírtak alapján ebben az esetben is kiszámolható a kör középpontja, O_1, majd az ütközési pontból, ami az ábrán F, a középpontba mutató vektor, ez $\vec{w}$. Most számoljuk ki a bejövő fénysugár, $\vec{v}$, és $\vec{w}$ által bezárt szöget. 0 és π közé esik. Az ábrán ez a szög γ, ami α kiegészítő szöge lenne.

A programban egy elágazással van megoldva, hogy a beesési szög értékét mindig azzal tegye egyenlővé, amelyik a két szög közül a kisebb, ez esetünkben α.

$$|\vec{v}| * |\vec{w}| \cos\gamma = \vec{v} * \vec{w} \rightarrow \gamma = \cos^{-1}\left(\frac{\vec{v} * \vec{w}}{|\vec{v}| * |\vec{w}|}\right)$$

Snellius-Descartes törvény:

$$\sin\alpha * n_1 = \sin\beta * n_2 \rightarrow \beta = \sin^{-1}\left(\frac{\sin\alpha * n_1}{n_2}\right)$$

A $\vec{v}$ irányát forgatással fogjuk megváltoztatni, két mátrixot használva, az egyik a pozitív, a másik a negatív irányba forgatja el a vektort. Esetfüggő, hogy melyikre van szükség. Most α – β szöggel kell elforgatni pozitív irányba.

A két mátrix: pozitív irányba $\rightarrow P \begin{bmatrix} \cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix}$, negatív irányba $\rightarrow N \begin{bmatrix} \cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$

Így $\vec{v'}$ re felírhatjuk:

$$\vec{v'} = \vec{v} \times P$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \begin{bmatrix} \cos(\alpha - \beta) & \sin(\alpha - \beta) \\ -\sin(\alpha - \beta) & \cos(\alpha - \beta) \end{bmatrix}$$

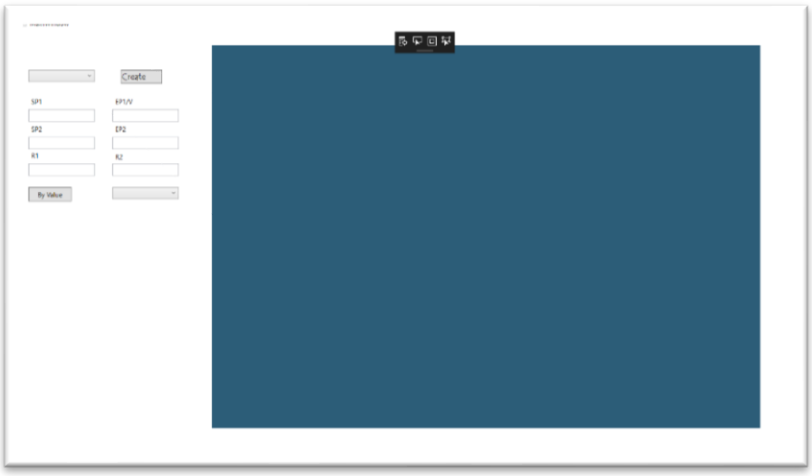
és ezzel megkaptuk a kívánt új irányvektort.

Az alkalmazás

Az alkalmazásban a legtöbb dolog megfelelően működik, viszont előfordulnak hibák, főleg a lencsék kapcsán. Ezek továbbfejleszthető, javítható problémák, amik orvoslására már nem maradt elegendő idő. Az alkalmazás nyelve angol, ez megkönnyíti a megírását, kommentelését. mivel a programozási nyelv is angol.

I. A felhasználó szemszögéből *(8. ábra)*

A kezelőfelület meglehetősen egyszerű: hat szövegdobozba adatokat lehet megadni, két gombbal lehet az alkalmazással kommunikálni és két lenyitódó panellel különböző opciók közül lehet választani. Mindenfajta kirajzolás a kék panelra történik, annak a bal felső sarka a koordinátarenszer 0;0 pontja.



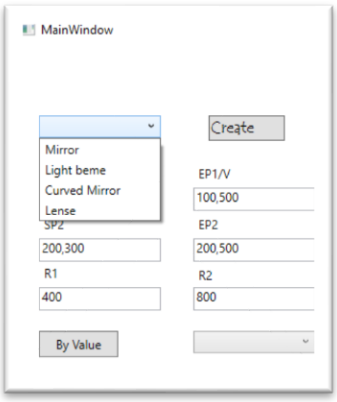
8. ábra.
GUI

Különböző eszközök/fénysugár hozzáadása

<u>Tükör</u>

Tükor hozzáadásához a felhasználónak a felső, lenyíló panelben ki kell választania a Mirror mezőt majd a Create gombra kattintani. Most, hogy kiválasztotta, mit kíván létrehozni, azt két módon teheti meg:

1. Az egér bal gombját megnyomja a kék panelen és azt lenyomva tartja. Ahol lenyomta, ott lesz a tükör egyik végpontja. A kurzort ezután a panelen mozgatva tetszőleges helyen fölengedi a gombot, a fölengedés helye lesz a tükör másik végpontja.
2. Az SP1 és EP1/V mezőbe beírja két pont koordinátáit. A pont megadásának formája: 43,32
   A koordináták beírása után a By Value gombra kattintva létrehozza a tükröt.



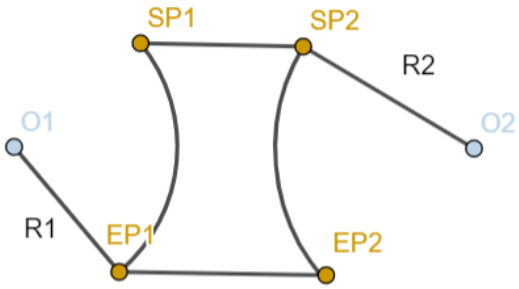
9. ábra.
A kívánt elem kiválasztása.

Újabb eszköz létrehozásához megint választani kell, a Create gombra kattintani, és a megfelelő adatokat megadva majd a ByValue gombra kattintva, vagy a kurzorral, létrehozható a kiválasztott tárgy.

A létrehozandó eszköz kiválasztása minden esetben ugyanaz, panelben választani aztán a Create gombra kattintani, így ezentúl csak a létrehozás részéről írok. *(9. ábra)*

Gömbtükör

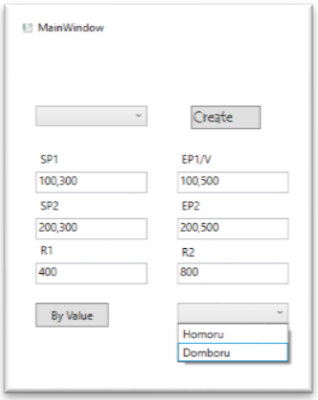
A létrehozása nagyon hasonló az egyszerű tükköréhez, annyi a különbség, hogy a felhasználó akár adatok megadásával, akár kurzorral kívánja létrehozni, az R1 mezőben meg kell előtte adni a görbületi sugár nagyságát. Innentől vagy megadja a két végpont koordinátáit és a By Value gombra kattint, vagy kurzorral teszi azt meg a fent leírtak alapján. Az, hogy a tükör milyen irányba néz, azon múlik, melyik az első megadott pont. Az első pont körül, az óramutató járásával ellentétes irányba fog "kidomborodni".



10. ábra.

Lencse

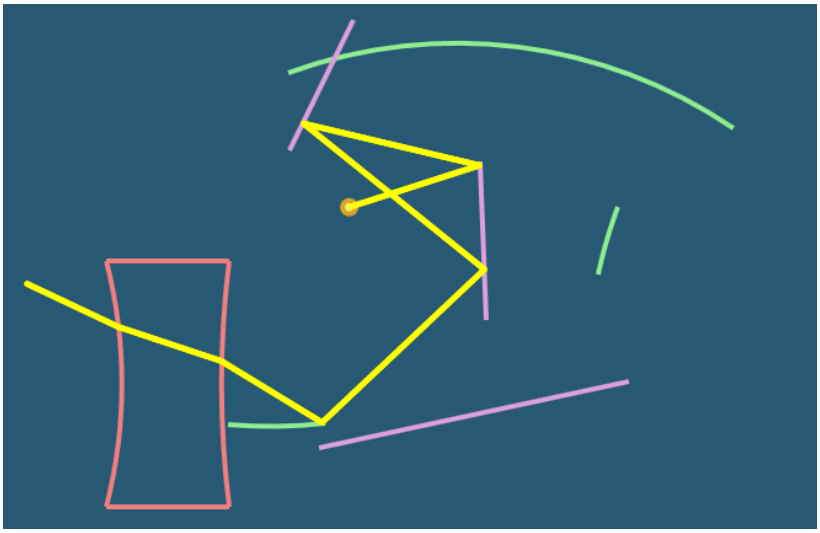
Lencse létrehozása egyedül adatokkal lehetséges, az ábráról leolvasható, hogy melyik adatmező minek felel meg *(10. ábra).* Az adatok beírása után a második panelben ki lehet választani, hogy homorú vagy domború lencsét kíván a felhasználó létrehozni. *(11. ábra)* Ez után a By Value gombra kattintva létrejön a lencse.



11. ábra.

Fénysugár

A fénysugár hozzáadása ugyancsak a tükköréhez hasonlít. Miután kiválasztottuk, kurzorral létrehozhatjuk úgy, mint egy egyenest, annyiban tér el, hogy most a lenyomás helyéről indul a sugár, abba az irányba, ahol fölengedtük a gombot. Ezzel tulajdonképpen megadjuk a fény irányát és a pontot, ahonnan indul. Adatok alapján is létrhozható, ekkor SP1 mező tartalma a kezdőpontja, EP1/V mező tartalma pedig az irányvektor, amivel elindul. A fénysugár sebességét jelenleg nem lehet változtatni a felhasználónak, az irányvektor változása miatt a sebessége minimálisan változik ütközések esetén,



12. ábra.
Az alkalmazás.

a közeg törésmutatója viszont nincs rá kihatással.

A fénysugár maga mögött húz egy nyomvonalat, így szemléletesebb például, amikor irányt változtat. *(12. ábra)*

II. A program

Az alkalmazás megírásához a C# programozási nyelvet választottam, azon belül WPF applikáció formájában készült el. A kód hossza (~1000sor) annak részletes ismertetését néhány oldalban nem teszi lehetővé, ezért inkább nagyvonalakban leírom hogyan működik, melyik része miért felelős. A projektet objektum orientáltan készítettem el és letölthető a következő címről: https://drive.google.com/open?id=1CSwEOfgjLL4HmR8eF3WCExmkm88aobUO

Fő működési elv

Miután a felhasználó elhelyezte a különböző elemeket és végül elindít egy fénysugarat, elindul egy *időzítő.* Az *időzítő* 3 milliszekundumonként kirajzol egy kört úgy, hogy ennek a körnek a középpontja minden kirajzolásnál odébb kerül, méghozzá a fényhez tartozó vektor nagyságával. Ez a vektor úgy van méretezve, hogy az egyik elem 1/-1 a másik pedig nagyobb mint -1 de kisebb, mint 1. A másik dolog, amit az időzítő csinál nem más, mint hogy megvizsgálja, hogy a kör középpontjának új koordinátái megegyeznek-e bármely lerakott elem valamely koordinátájával. (Azért fontos, hogy a fény irányvektora ilyen kicsi legyen, mert egyébkénet lehetséges lenne, hogy "átugrik" például egy tükröt.) Ha megegyezik, akkor először az a függvény kerül meghívásra, ami az ütközés eseménykezelője. Ez a függvény "eldönti", hogy melyik elemmel ütközött, majd meghívja annak az elemnek az ütközés függvényét, ami végül az elméleti részben leírtak alapján megváltoztatja a fény irányvektorát. A fénnyaláb ezután új irányba megy tovább és újabb ütközés esetén ugyanez megismétlődik.

Kicsit részletesebben

Objektum orientáltan készült az alkalmazás, ezért először bemutatom a különböző osztályokat.

13. ábra.

Az első *MainWindow.xaml.cs - be* kerül az a kód, amit szeretnék, hogy végrehajtson a számítógép. A *MainWindow.xaml – ben* lehet dizájnolni az alkalmazást, én ezt kevésbé használtam. A *Mirror.cs* az az osztály, ami, mint tükör funkcionál, az osztályon belül két *Point* változó van, a tükör két végpontjának. A *Curved_mirror.cs* gömbtükörként viselkedik, a *Mirror.cs*-ből örököli a két pontot és mellette még van egy változó, ami a tükör sugarának nagyságát tárolja. A *Lense.cs* 6 adatot tárol, ez az osztály reprezentálja

a lencsét. A 6 adat a tizedik ábránál már említve volt. A *Light.cs* a fény irányvektorát tárolja, valamint a függvényt, ami azt átméretezi a -1 – 1 tartományra. A *Matrix.cs* egy adatszerkezetként működik, mindössze egy 2x2 – es mátrix elemeit tárolja. A *Vector.cs* is adatszerkezet, ami egy kételemű vektor elemeit tárolja, a *Light.cs* osztályon belül is egy ilyen vektorban van eltárolva a fény irányvektora. A *Matrix_operation_static.cs* egy statikus osztály, aminek olyan függvényei vannak, amelyek vektorokkal és mátrixokkal kapcsolatos műveleteket hajtanak végre.

Mindent duplán tárolni?

Amikor a felhasználó létrehoz egy tükröt, azt grafikusan egy *Line* objektum formájában jeleníti meg a számítógép két végpontja alapján. Ezt a *Line* objektumot aztán elmentem egy *Listába.* Amikor a *Timer* egyezést talál egy elem és a fény pontja között, véigigviszgálja a *Line objektumok Listáját*, és amikor egyezést talál, tudjuk, hogy melyik objektummal ütközött a fény. Viszont ennek a *Line objektumnak* már nem férünk hozzá a két végpontjához, amik pedik szükségések ahhoz, hogy kiszámoljuk az új irányvektort. Ezt úgy oldom meg, hogy amikor létrehozok egy *Line objektumot,* létrehozok egy *Mirror objektumot* is, amit aztán egy *Mirror Listába* mentek. Ez a *Mirror objektum* tárolja a tükör két végpontját. Így amikor tudjuk, melyik *Line objektummal* ütközött a fény, annak tudjuk a sorszámát a listában. A *Mirror listában* ugyanilyen sorszámmal megtalálható az a *Mirror objektum,* ami tárolja a tükör két végpontjának koordinátáit és ezzel már kiszámolható a fény új irányvektora. Nagyon hasonlóan van az adatok tárolása megoldva a gömbtükrök és a lencsék esetében is.

```
private void create_mirror()
{
    //create the line
    Line new_line = new Line()
    {
        X1 = s_Point_1.X,
        Y1 = s_Point_1.Y,
        X2 = e_Point_1.X,
        Y2 = e_Point_1.Y,
        Stroke = Brushes.Plum,
        StrokeThickness = 4,
    };
    //display the line to the canvas
    cnv_display.Children.Add(new_line);

    //add the line to the lines List
    lines.Add(new_line);
    //add the line to the mirrors List
    mirrors.Add(new Mirror(s_Point_1, e_Point_1));

    mirror_create = false;
}
```

14. ábra.
Tükör létrehozása.

Eszközök létrehozása

Amikor létrehozunk egy eszközt, a programon belül egy függvény kerül meghívásra, attól függ melyik, hogy milyen elemet hozunk létre *(15. ábra)*. Ennek a függvénynek az a feladata, hogy beolvassa a szükséges adatokat, valamint, a tükör esetében például létrehozza a Mirror és Line objektumokat, és ezeket hozzáadja a megfelelő listához *(14. ábra)*.

```
private void create_mirror()...

private void create_curved_mirror()...

private void create_lense()...
```

15. ábra.

Befejezésül

Az alkalmazás működőképes, továbbfejleszthető a meglévő hibák kijavításával, valamint további eszközök hozzáadásával, a fénysugár sebességének realisztikus változásával, a kód optimalizásával.

Felhasznált háttéranyagok

Kapcsolódó témák a következő weblapokról:

- https://stackoverflow.com/
- https://docs.microsoft.com/en-us/dotnet/csharp/

Tartalomjegyzék