« بسم الله الرحمن الرحیم »

[1. مقدمه](#_Toc472855947)

[2. کتاب خانه های استفاده شده](#_Toc472855948)

[3. الگوریتم پیاده سازی شده](#_Toc472855949)

[3. 1 ایجاد توپ ها](#_Toc472855950)

[3. 2 حرکت توپ ها](#_Toc472855951)

[3. 3 کشیدن توپ ها](#_Toc472855952)

[3. 4 داخل صفحه نگه داشتن توپ ها](#_Toc472855953)

[3. 5 تداخل توپ های متحرک با توپ های بزرگ شده](#_Toc472855954)

[3. 6 بزرگ و کوچک شدن توپ ها](#_Toc472855955)

[3. 7 کلیک کردن](#_Toc472855956)

[3. 8 اتمام مرحله](#_Toc472855957)

[3. 9 مدیریت حافظه](#_Toc472855958)

[4. توابع تعریف شده](#_Toc472855959)

[4. 1 ball()](#_Toc472855960)

[4. 2 ball(float x , float y)](#_Toc472855961)

[4. 3 void keepInside()](#_Toc472855962)

[4. 4 void action()](#_Toc472855963)

[4. 5 int haveIntersect(Ball\* const b1 , Ball\* const b2)](#_Toc472855964)

[4. 6 int checkIntersection(ball\* b , vector<Ball> & extended\_balls)](#_Toc472855965)

سید علی میرفردوس

پروژه پایان ترم مبانی کامپیوتر: پیاده سازی بازی بوم شاین

استاد: دکتر نصیحت کن

زمستان 1395

[4. 7 void drawBalls(vector<Ball> const & balls, vector<Ball> const & extended\_balls)](#_Toc472855966)

[4. 8 void moveBalls(vector<Ball> & balls, vector<Ball> & extended\_balls , int \* wonballs)](#_Toc472855967)

# 1. مقدمه

بازی بوم شاین بدین شکل انجام می شود که با کلیک کردن روی صفحه، یک توپ شروع به بزرگ شدن می کند تا به یک شعاع خاص برسد؛ آنگاه کوچک می شود تا از صفحه محو گردد. در طی این فرآیند، هر توپ متحرک دیگری که به آن برخورد کند نیز شروع به بزرگ شدن می کند.

در هر مرحله بازیکن باید بتواند حداقل به تعدادی خاص، توپ ها را بزرگ کند. بازی شامل 12 مرحله است که به ترتیب سخت تر می شوند و در هر مرحله تعداد توپ های مورد نیاز افزایش می یابد.

در این پروژه قصد داریم این بازی را به صورت برنامه ی کنسولی (Console Application) همراه با کتابخانه گرافیکی SFML پیاده سازی کنیم.

# 2. کتاب خانه های استفاده شده

در این پروژه کتابخانه های زیر استفاده شده است:

1. stdafx.h
2. SFML/Graphics.hpp: کتابخانه گرافیکی
3. SFML/System.hpp: ماژول پایه ی SFML
4. SFML/Window.hpp: ماژول مربوط به کلیک کردن و کشیدن پنجره ی گرافیکی
5. stdlib.h: برای استفاده از توابع srand, rand
6. time.h: برای استفاده از تابع time
7. vector: برای ایجاد وکتوری از توپ ها
8. iostream: برای استفاده از تابع cout

# 3. الگوریتم پیاده سازی شده

## 3. 1 ایجاد توپ ها

یک استراکت به نام توپ تعریف شده که شامل کلاس دایره ی SFML و تعدادی متغیر و تابع است که توضیحات هر کدام در ادامه می آید.

دو وکتور از توپ ها ایجاد می شوند:

1. توپ هایی است که بزرگ شده اند یا در حال بزرگ و کوچک شدن هستند و دیگر حرکت نمی کنند.
2. توپ های متحرک

در ابتدای شروع هر مرحله، وکتور توپ ها با توپ هایی که هر کدام به وسیله ی تابع ایجاد کننده (Constructor) مربوطه با یک رنگ و موقعیت تصادفی (Random) ایجاد شده اند، پر می شود.

متغیر (wonballs) تعداد توپ هایی است که کاربر توانسته بزرگ کند و در ابتدای شروع هر مرحله مقدار صفر دارد.

در هر مرحله تعداد توپ ها با فرمول ( مرحله × 5 ) محاسبه می شود.

## 3. 2 حرکت توپ ها

وکتورهای مربوطه به تابعی به نام (moveBalls) حرکت داده می شوند.

هر توپ به عنوان نمونه ای از استراکت «توپ»، دارای تابعی به نام (move) می باشد که تابع تعیین موقعیت SFML، تابعی مربوط به داخل صفحه نگه داشتن توپ و تابع تعیین شعاع را صدا می زند.

تابع تعیین موقعیت، موقعیت فعلی توپ را با مقدار دقت حرکت توپ ها جمع کرده و آن را به جای موقعیت فعلی قرار می دهد.

سپس به ازای حرکت مربوطه را بررسی می شود که آیا منجر به تداخل توپ حرکت کرده و یکی از توپ های بزرگ شده موجود شده یا نه.

در صورت وجود تداخل، تعداد توپ های برنده شده را یکی افزایش داده و سپس مولفه سرعت افزایش شعاع را برابر 1 قرار می دهد.

در پایان بزرگ شدن نیز زمان انتهای فرآیند را در متغیر مربوط توپ ذخیره می کند تا توپ بعد از گذشت زمان مشخص، شروع به کوچک شدن کند.

پس از حرکت دادن توپ ها به روش بالا، تابع کشیدن توپ ها صدا زده می شود.

## 3. 3 کشیدن توپ ها

وکتورهای مربوطه به تابعی به نام (drawBalls) داده می شوند که وظیفه کشیدن گرافیکی آن ها را دارد.

ابتدا صفحه گرافیکی پاک می شود؛ سپس تمام توپ های متحرک و بعد تمام توپ های در حال بزرگ شدن کشیده می شوند. در نهایت، توپ های کشیده شده نمایش داده می شوند و مدت زمان کوتاهی نیز توقف ایجاد می شود تا حس حرکت نرم توپ ها به وجود آید.

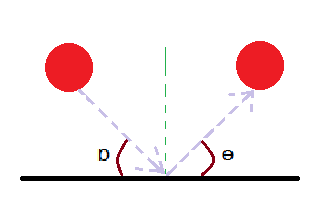
## 3. 4 داخل صفحه نگه داشتن توپ ها

هر توپ به عنوان یک نمونه از استراکت «توپ» تابعی به نام (keepInside) دارد که این امر را به انجام می رساند.

این تابع طول و عرض موقعیت فعلی توپ را با مقدار دقت حرکت جمع می کند و چک می کند که آیا این مقدار داخل صفحه هست یا نه. الگوریتم چک کردن به شکل زیر است:

* چپ صفحه: طول به دست آمده، بزرگ تر از صفر باشد.
* راست صفحه: طول به دست آمده، کم تر از طول بیشینه صفحه منهای قطر توپ باشد.
* بالای صفحه: عرض به دست آمده، بزرگ تر از صفر باشد.
* پایین صفحه: عرض به دست آمده، کم تر از عرض بیشینه صفحه منهای قطر توپ باشد.

چنانچه توپ به لبه بالا یا پایین صفحه رسیده باشد، عرض آن در یک 1- ضرب می شود تا جهت حرکت در راستای عرض تغییر کند.

همین فرآیند به طور مشابه برای طول ها نیز تکرار می شود.

تحلیل این الگوریتم به طور فیزیکی بدین شکل است:

با فرض ثابت ماندن سرعت داریم: که باعث برابری تک تک مولفه می شود. پس از ساده سازی داریم:

که نتیجه می شود:

اولا ،

دوما الگوریتم بالا اثبات می شود.

## example3. 5 تداخل توپ های متحرک با توپ های بزرگ شده

اگر فرض کنیم دو دایره با شعاع های r0 و r1 تداخل دارند، در نقاط p3 متقاطع هستند که یا دو نقطه یا یک نقطه می باشد.

در شکل رو به رو داریم: http://www.csharphelper.com/howto_circle_circle_intersection_eq1.png

که نشان دهنده فاصله بین مراکز دو دایره است.

با محاسبه ی http://www.csharphelper.com/howto_circle_circle_intersection_eq2.png داریم:

* d > r0 + r1: فاصله دایره ها از هم زیاد است و هیچ تداخلی وجود ندارد.
* d < |r0 – r1|: یکی از دایره ها داخل دیگری است و در نتیجه هیچ تداخلی وجود ندارد.
* d = 0 یا r0 = r1: دو دایره یکی هستند.
* d = r0 + r1: دو دایره در یک نقطه تقاطع دارند.
* اگر هر کدام از حالت های بالا نبود، دو دایره در دو نقطه متقاطع هستند.

## 3. 6 بزرگ و کوچک شدن توپ ها

برای هر توپ یک مولفه سرعت تغییر شعاع با نام dr در نظر گرفته شده که در ابتدا برای همه ی توپ ها برابر 0 است.

وقتی توپ شروع به بزرگ شدن می کند، این مقدار برابر با 1 می شود.

بعد از رسیدن به مقدار مورد نظر، مقدار صفر را می گیرد و در ضمن، زمان آن واقعه در متغیری ذخیره می شود.

در هر بار صدا زدن تابع حرکت، زمان فعلی با زمان پایان فرآیند بزرگ شدن چک می شود و اگر بزرگ تر از مقدار مورد نظر بود، مقدار dr برابر 1- قرار می گیرد تا توپ شروع به کوچک شدن کند.

وقتی شعاع توپ به صفر رسید، از لیست توپ ها حذف می شود.

## 3. 7 کلیک کردن

به محض کلیک کردن روی صفحه، تابع ایجاد کننده (Constructor) صدا زده می شود و توپی با مختصات محل موس و با رنگی تصادفی ایجاد شده و در لیست توپ های بزرگ شده قرار می گیرد. سرعت افزایش شعاع اولیه این توپ برار 1 در نظر گرفته می شود.

متغیری به نام clicked که در ابتدای شروع هر مرحله برابر 0 تعریف شده بود، برابر 1 می شود تا دیگر اجازه ی کلیک کردن مجدد به کاربر ندهد.

## 3. 8 اتمام مرحله

در صورتی که کلیک انجام شده بود و دیگر توپی در حال بزرگ و کوچک شدن در لیست نمانده بود، مرحله تمام می شود؛ سپس تعداد توپ هایی که کاربر توانسته بزرگ کند با مقدار مورد نیاز مقایسه می شود و پیام مناسب نمایش داده می شود.

## 3. 9 مدیریت حافظه

در پایان هر مرحله وکتورها تخلیه می شوند تا حجم کمتری حافظه هدر رود.

# 4. توابع تعریف شده

## 4. 1 ball()

تابع ایجاد کننده توپ ها که درون استراکت تعریف شده و بدون گرفتن هیچ متغیری، موقعیت، سرعت حرکت و رنگ توپ را به صورت تصادفی و شعاع توپ را برابر شعاع پیش فرض قرار می دهد.

## 4. 2 ball(float x , float y)

تابع دیگر ایجاد کننده توپ ها که درون استراکت تعریف شده و هنگامی که کاربر کلیک می کند، توپی با این تابع ایجاد می شود.

این تابع موقعیت توپ را به عنوان ورودی می گیرد؛ رنگ توپ را به صورت تصادفی و شعاع توپ را برابر شعاع پیش فرض قرار می دهد؛ سرعت تغییر شعاع را برابر 1 و سرعت حرکت توپ را برابر با 0 تعریف می کند.

## 4. 3 void keepInside()

این تابع چک می کند که اگر توپ به لبه های صفحه رسیده، موقعیت آن را طوری تغییر دهد تا مجددا به درون صفحه بازگردد.

این تابع هیچ ورودی و خروجی ندارد؛ توسط تابع حرکت صدا می زند؛ درون استراکت تعریف شده است.

## 4. 4 void action()

این تابع وظیفه حرکت توپ را دارد که هم موقعیت آن را بر اساس موقعیت قبلی و سرعت حرکت تغییر می دهد و هم شعاع آن را بر اساس سرعت تغییر شعاع مشخص می کند؛ چک کردن های مربوط به تغییرات سرعت شعاع نیز توسط این تابع انجام می شوند.

تابع درون صفحه نگه داشتن توپ، توسط این تابع صدا زده می شود.

این تابع هیچ ورودی و خروجی ندارد؛ توسط تابع حرکت همه توپ ها صدا زده می شود؛ درون استراکت تعریف شده است.

## 4. 5 int haveIntersect(Ball\* const b1 , Ball\* const b2)

این تابع وجود تداخل بین دو توپ را بر اساس الگوریتم بالا بررسی می کند.

توپ ها به صورت By Reference به این تابع داده می شوند و با توجه به این که const تعریف شده اند، هیچ تغییری روی توپ ها به وجود نمی آید.

در صورت وجود تداخل عدد 1 و در غیر این صورت عدد 0 بازگردانده می شود.

## 4. 6 int checkIntersection(ball\* b , vector<Ball> & extended\_balls)

این تابع وجود تداخل بین یک توپ خاص و تمام توپ های بزرگ شده را به وسیله تابع قبل بررسی می کند.

در صورت وجود اولین تداخل عدد 1 و در صورتی که هیچ تداخلی وجود نداشت، عدد 0 بازگردانده می شود.

## 4. 7 void drawBalls(vector<Ball> const & balls, vector<Ball> const & extended\_balls)

این تابع وظیفه کشیدن و نمایش توپ ها را بر عهده دارد. وکتوری از توپ های در حال حرکت و توپ های بزرگ شده به آن داده شده است که اولا به صورت ثابت و دوما پوینتر می باشند.

در پایان نمایش، مدت کوتاهی، برنامه را متوقف می کند تا حس حرکت نرم را ایجاد کند.

این تابع مقداری بر نمی گرداند و توسط تابع حرکت همه ی توپ ها صدا زده می شود.

## 4. 8 void moveBalls(vector<Ball> & balls, vector<Ball> & extended\_balls , int \* wonballs)

این تابع وظیفه حرکت دادن همه توپ ها را بر عهده دارد. وکتوری از توپ های در حال حرکت، توپ های بزرگ شده و تعداد توپ هایی که تا به حال برده شده به آن داده می شود؛ سپس همه ی توپ های در حال حرکت را حرکت داده و ایجاد تداخل را نیز چک می کند؛ آنگاه بزرگ و کوچک شدن توپ ها را بر عهده می گیرد و در نهایت، تابع کشیدن توپ ها را صدا می زند.

این تابع مقداری بر نمی گرداند.