فرزان مسیبی -۴۰۰۵۲۲۱۷۵-

بازی زندگی

منطق کلی:

در ابتدا یک آرایه دوبعدی ۱۰۰ در ۱۰۰ با نام table، به عنوان جدول اصلی بازی تعریف کردیم که فرآیند های اصلی برنامه برروی این آرایه اعمال خواهند شد. سپس با استفاده از تابع initialize() سلول هایی را که در مرحله اول زنده خواهند بود بطور تصادفی در آرایه پخش کردیم. (سلول های زنده با عدد ۱ و سلول های مرده با عدد ۰ نمایش داده میشوند.)

تابع initialize() :

طرز کار این تابع به گونه ای است که تا رسیدن به تعداد تصادفی که در ابتدای برنامه تعریف شده، در هر مرحله یک مختصات تصادفی تولید میکند؛ یکی برای سطر و یکی برای ستون که مقدار هردو، عددی از صفر تا ۹۹ میباشد. حال اگر سلول با این متخصات زنده باشد یا به عبارتی عددش ۱ باشد، برنامه مختصات جدیدی تولید کرده و آزمایش میکند، و اگر این سلول مرده باشد یا به عبارتی عددش ۰ باشد، برنامه، مقدار آن را با عدد ۱ جایگزین میکند.

بعد از موارد بالا، دو آرایه دوبعدی دیگر با نام های dead_cells و healed_cells به ابعاد ۱۰۰۰۰ در ۲ و همچنین دو متغیر با نام های dead_index و healed_index با مقدار های اولیه صفر تعریف کردیم.

آرایه dead_cells حاوی مختصات سلول های زنده ای است که در مرحله بعد مرده خواهند بود و همچنین آرایه healed_cells نیز حاوی مختصات سلول های مرده ای است که در مرحله بعد دوباره زنده خواهند شد.

موارد بالا، ساختار کلی برنامه را مشخص میکنند.

فرآیند اصلی برنامه به اینصورت است که در یک حلقه while که میتواند به تعداد مراحل مشخص شده یا به تعداد بی نهایت بار پیش برود، برنامه هربار تعداد همسایه های تک تک خانه(سلول) های آرایه table را بوسیله تابع alives_around() محاسبه کرده و با توجه به قوانین بازی، مختصات هر سلول زنده ای را که در مرحله بعد مرده خواهد بود، در خانه با شماره dead_index آرایه dead_cells ذخیره کرده و یک واحد به dead_index اضافه میکند. بنابراین، در اینجا مختصات سلول در خانه اول آرایه ذخیره شد و مختصات سلول بعدی در خانه بعدی آرایه ذخیره خواهد شد. این مورد برای خانه های مرده ای که در مرحله بعد زنده خواهند شد نیز به همین صورت است.

طریقه ذخیره کردن مختصات نیز به این صورت است که شماره سطر سلول در مرتبه اول خانه با شماره dead_index آرایه dead_cells ذخیره شده و شماره ستون آن نیز در مرتبه دوم ذخیره می شود. این عمل برای آرایه healed_cells نیز به همین صورت است.

پس از آنکه همه سلول ها به صورتی که در بالا ذکر شد بررسی شدند، برنامه، مقدار سلول هایی را که مختصاتشان در خانه با شماره dead_index آرایه dead_cells ذخیره شده، به ۰ تغییر داده و از مقدار dead_index یک واحد کم میکند و مقدار خانه هایی را که مختصاتشان در خانه با شماره healed_index آرایه healed_cells ذخیره شده به ۱ تغییر می دهد و از مقدار healed_index یک واحد کم میکند. پس از اینکه این فرایند روی تمامی مختصات های ذخیره شده اعمال شد، مقدار dead_index و healed_index هردو دوباره برابر با صفر شده و برنامه به مرحله بعد میرود.

ابزار ها و توابع مورد استفاده:

برای تبدیل این برنامه به برنامه ای گرافیکی، از کتابخانه(SFML(Simple and Fast Multimedia Library استفاده شده است.

در دستور های مربوط به این کتابخانه از فضای نامی sf که مختص به همین کتابخانه است استفاده میکنیم.

با استفاده از کلاس RenderWindow پنجره مورد نظر را با نام window ایجاد کرده و با استفاده از تابع VideoMode طول و عرض آن را تعریف کرده و عنوان را تعریف میکنیم. ولی این پنجره پس از باز شدن، بلافاصله بسته خواهد شد. برای جلوگیری از این اتفاق از یک حلقه while با تابع window.isOpen() به عنوان ورودی کمک خواهیم گرفت که حلقه while اصلی برنامه در داخل آن قرار خواهد گرفت.

سپس یک Image ۴۰۰ در ۴۰۰ با نام image تعریف میکنیم. Image در واقع آرایه ای دوبعدی از پیکسل هاست.

در این Image برنامه ما ۴ تا ۴ تا جلو میرود؛ یعنی درواقع یک جدول ۱۰۰ در ۱۰۰ با ۱۰۰۰۰ مربع داریم که طول ضلع هر مربع برابر با ۴ پیکسل خواهد بود.

تابع بعدی که از آن استفاده شده است، تابع image.setPixel() است که از آن برای تعیین رنگ هر پیکسل با دادن شماره سطر و ستون و رنگ به عنوان ورودی، رنگ پیکسل مورد نظر را تغییر میدهد.

با استفاده از حلقه for و این تابع، در هر مرحله خطوط اطراف جدول را با رنگ خاکستری، مربع های مرده را با سیاه و مربع های زنده را با رنگ سبز مشخص خواهیم کرد.

وارد حلقه while(window.isOpen()) می شویم. با آغاز فرآیند اصلی برنامه، ترکیب برنامه آرایه و موارد گرافیکی ذکر شده به اینصورت است که، مختصات هر سلول در آرایه table به مختصات پیکسل گوشه بالای سمت چپ مربع های فرضی در image تبدیل میشود که با حلقه for و تابع image.setPixel() با شروع از آن نقطه رنگ کل مربع فرضی را مشخص میکنیم.

یعنی برای تک تک پیکسل ها رنگ مشخص میکنیم و در آخر image بصورت یک جدول مربع مربع دیده خواهد شد.

سپس یک Texture با نام texture برای بارگذاری image در آن و سپس یک Sprite با نام sprite برای بارگذاری texture در آن تعریف میکنیم.

image را با استفاده از تابع texture.loadFromImage(image) در texture و texture را در هنگام تعریف sprite در آن بارگذاری میکنیم.(sf::Sprite sprite(texture))

سپس در هر مرحله، پنجره را با استفاده از تابع window.clear() پاک کرده و با استفاده از توابع window.draw(sprite) و window.display() و مرحله جدید نمایش داده میشود و برنامه به مرحله بعدی میرود.