Aleksandra Wyszyńska

Link do repozytorium: https://github.com/AWyszynska/Projekt\_C\_-.git

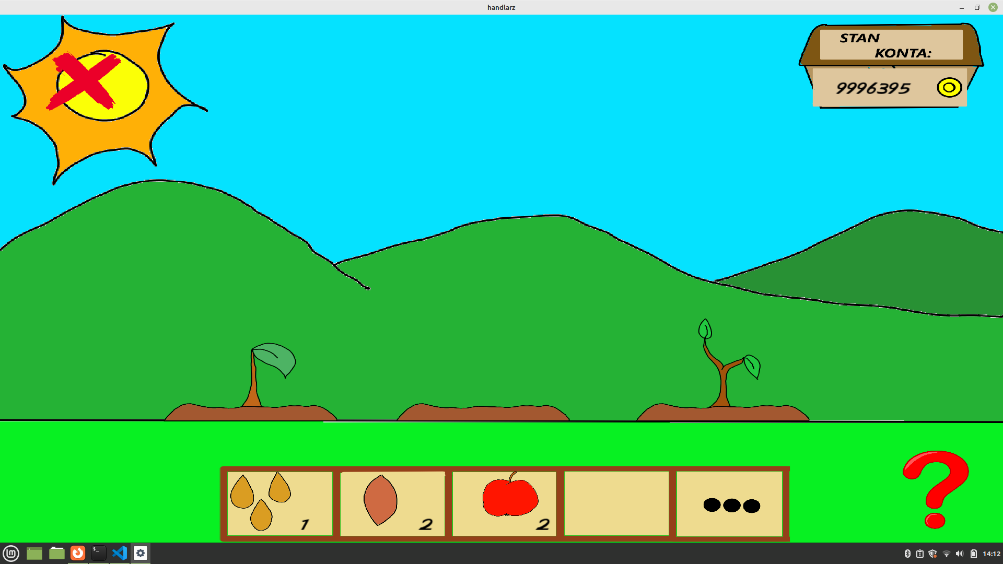
HANDLARZ

Moja gra pt. ,,handlarz” została napisana za pomocą biblioteki SFML. Wizja początkowa różni się od wizji końcowej co jest niestety spowodowane u mnie brakiem czasu. Na początku miałam na celu stworzenie gry, w której wykonuje się zadania na podstawie zebranych materiałów albo zdobytych pieniędzy. Ile zadań by się zrobiło tyle poziomów można by było zdobyć. Gra miała polegać na zbieraniu materiałów w innych miejscach a potem sprzedawaniu ich w sklepie co udało mi się wykonać. Mój pomysł na stworzenie takiej gry wziął się od prostych gier z dzieciństwa, kiedy sadziło się zbierało, kupowało itd. Stwierdziłam, że chciałabym sama zrobić coś takiego.

Udało mi się stworzyć dwie klasy ogród i sad, w których gracz może posadzić nasiona kupione w sklepie i użyć ich do posadzenia. Tutaj moja wizja się najlepiej moim zdaniem udała, chodź te klasy zajęły mi najwięcej czasu. Nasiona rosną z upływem czasu nawet podczas zamknięcia strony albo przemieszczania się między innymi klasami, do czasu w którym już roślina urosła dodaje czas który przez ten czas zleciał, dzięki temu roślina podczas zamknięcia strony cały czas ,,rosła’’. Mój sposób wykonania polegał na tym że po każdym posadzeniu, każda roślina stawała się strukturą gdzie zapisywane jest struktura początkowa, czas który upływał, czy roślina jest posadzona oraz jej pozycja. Gdy roślinka jest zbierana dodawana jest do wektora jej pozycja a następnie sortowana za każdym razem dodawania nowej roślinki. Ma to na celu dawanie roślinom równomierną pozycje posadzenia, bez tego roślina wychodziłaby po za okno co jest spowodowane pętlą która dodaje nową roślinę co określoną odległość. Więc jeśli zwolni się miejsce w to samo miejsce albo wcześniejsze po sortowania zajmuje miejsce kolejna roślina. Również po zebraniu rośliny niszczona jest struktura w której występowała.

Podczas zamknięcia okna i otworzeniu go ponownie rośliny są na tych samych pozycjach co były przed zamknięciem i mają taki sam czas wzrostu. Dzięki temu, że rośliny, które są posadzone zapisuje w osobnym wektorze, które podczas zamknięcia strony zapisuje w pliku tak samo jak czas struktury i pozycje. Klasa ogród i sad działa na takiej samej zasadzie.

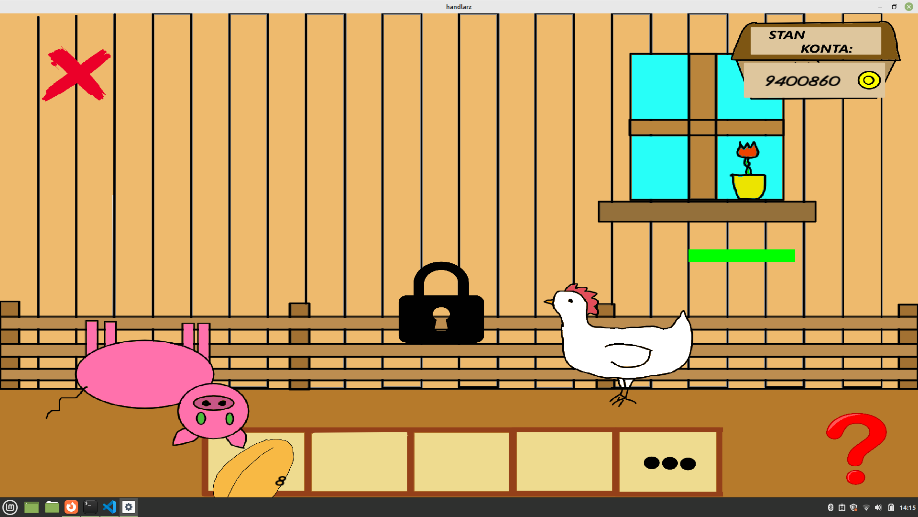




Klasa kopania działa w następujący sposób: po naciśnięciu spacji generuje się struktura, którą należy przytrzymać przez określony czas. Struktura w międzyczasie rośnie, a gdy czas dobiegnie końca do wektora z zapisywanymi strukturami dodaje się lodowo jeden z surowców. Surowiec który możesz wylosować jest określany za pomocą rand, za której pomocą generowałam losowe liczby od 0 do 100, które zamieniałam na procent. Niestety podczas generowania nowej struktury, zdjęcie kolejnej może ulec zniekształceniu i struktura jest na przykład w połowie przycięta. Zależy to od tego jaka struktura zostanie pierwsza wygenerowana, jeśli kolejne będą miały większą rozdzielczość to wtedy zostają ucinane, a jeśli mniejsze to są w normalnej formie. Kolejnym błędem w tej klasie jest to, że nie wiadomo z jakiego powodu ale pierwsza generowana struktura po naciśnięciu i przytrzymaniu przycisku nie rośnie. Niestety, w klasie kopalnia nie jest wywołana klasa, która jest odpowiedzialna za wyświetlanie dymków po najechaniu konkretnej struktury na pasku z powodu takiego, że skala po przytrzymaniu dochodziła do bardzo dużych rozmiarów i nie rosła co określony czas tylko bez przerwy. Z uwagi obaw przed niedziałającym programie postanowiłam jej tam nie wywoływać.



Klasa hodowla to klasa w której istnieje możliwość kupienia własnego zwierzątka. na samym początku strony możemy zobaczyć kłódki, które możemy odblokować posiadając konkretne jajko zwierzątka. Jeśli je posiadamy możemy je odblokować. Zwierzątko porusza się według osi x w jedną i drugą stronę, co 5 sekund zmieniając kierunek toru ruchu na przeciwny oraz zdjęcie na odwrócone odbiciem lustrzanym. Na górze każdego zwierzaka widać pasek życia który można przedłużyć kupując lub sadząc nasionko w ogrodzie. Pasek jest wyświetlany za pomocą klasy, która pobiera czas całkowity życia zwierzaka, długość paska (która przy zamknięciu i otworzeniu strony zostaje taka sama, dzięki podzieleniu czasu który został przez czas całkowity pomnożony przez długość początkowa paska),szerokość i kolor. Gdy nie zdążymy go nakarmić, umiera. Zwłoki możemy wyczyścić a na ich miejscu pojawia się od początku kłódka, którą na nowo trzeba odblokować. Niestety z powodu braku czasu nie udało mi się stworzyć tą klasę w taki sposób aby zwierzaki też generowały surowce co jakiś czas które również można by było sprzedać.



Wszystkie nasiona oraz jajka można kupić w sklepie który działa na prostej zasadzie, gdy naciśniesz na wybrany surowiec i będziesz chciał go kupić to ilość pieniędzy w prawym górnym rogu zmaleje o podaną kwotę, a zakup zostanie dodany do paska na dole. Jak nie będziesz miał w tylu pieniędzy ile potrzeba to wyświetli się komunikat mówiący o tym. Jeśli natomiast będziemy chcieli sprzedać swoje zbiory będzie to możliwe po naciśnięciu na wybrany obiekt prawym przyciskiem myszy. W moim zamyśle gry chciałam aby była możliwość posiadania własnego sklepu, gdzie można byłoby sprzedawać surowce za określoną przez siebie kwotę, chciałam zrobić to tak że im bardziej zawyżona cena produktu tym dłużysz czas oczekiwania na sprzedać. Produkty miały być stawiona na pólkach z wyznaczoną ceną. Niestety również z powodu braku czasu nie udało mi się tego zrobić.



W programie znajduje się również magazyn który jest otwierany za każdym razem gdy naciśniemy trzy kropeczki na końcu paska na dole. W tym miejscu możemy zamienić struktury miejscami klikając najpierw na jedną a potem na drugą.



Również znajduje się klasa odpowiedzialna za wyświetlanie dymków nad podaną strukturą, na dole paska. Klasa nie działa tak jak powinna ponieważ podczas działania innej klasy gdy zostanie dodany lub usunięty jakiś surowiec, zmiany się nie zapisują. Wcześniejszej napisanej klasie występował problem, że klasa spowalniała program, albo gdy zbyt dużo się działa to zamykała go.

