C++ Project

Αναφορά Εργαστηριακής Άσκησης
Σπύρος Καφτάνης ΑΜ 5542
2013-2014

Περιεχόμενα

Εισαγωγή

Διάγραμμα Κλάσεων

Η Κλάση Simulate και το GameLoop

Οθόνη έναρξης και μενού

Βασικό περιβάλλον/συναρτήσεις & συμβάσεις

Εισαγωγή Ρομπότ

Συναρτήσεις κίνησης των ρομπότ

Συναρτήσεις λειτουργίας των ρομπότ

Επεξεργασία στοιχείων του χάρτη & bash script

Αφαίρεση ρομπότ & bash script

Εμφάνιση στοιχείων του χάρτη

Παράμετροι Εκτέλεσης του Προγράμματος

Βιβλιογραφία

Εισαγωγή

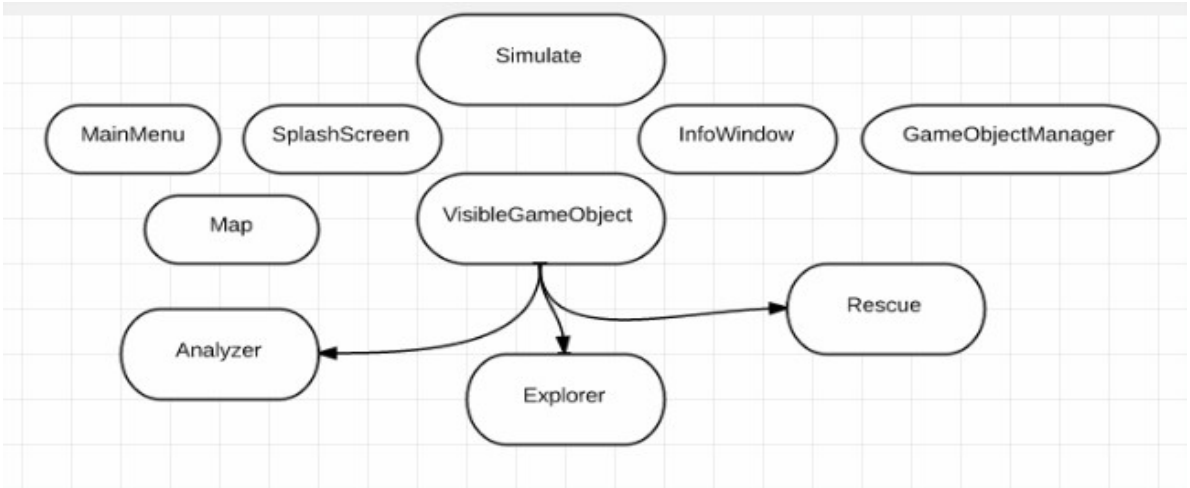
Το project αυτό δημιουργήθηκε με τη βοήθεια της βιβλιοθήκης γραφικών SFML 2.1 στη γλώσσα προγραμματισμού C++. Για τη συγγραφή του χρησιμοποιήθηκε το *Code::Blocks 12.11* με compiler τον *GCC* στην έκδοση 4.8.2.

Το λειτουργικό σύστημα το οποίο χρησιμοποιήθηκε ήταν η διανομή GNU/LINUX, Fedora 20 64bit με Linux Kernel στην έκδοση 3.12.x, στα γραφικά περιβάλλοντα Gnome, KDE και XFCE.

Εκτός από τη C++, υλοποιήθηκαν δύο bash scripts για τις λειτουργίες της επεξεργασίας και της αφαίρεσης, έτσι ώστε να μην χρειαστεί η χρήση κονσόλας για την εισαγωγή δεδομένων. Όπως θα δείτε και παρακάτω, τα scripts αυτά χρησιμοποιούνται μόνο σαν μεσολαβητής και σε καμία περίπτωση δεν εκτελούν κάποιες άλλες λειτουργίες.

Διάγραμμα Κλάσεων

Το παρακάτω διάγραμμα παρουσιάζει τις σχέσεις κληρονομικότητας μεταξύ των κλάσεων. Τέτοια σχέση υπάρχει μεταξύ της κλάσης VisibleGameObject και των τριών κλάσεων που περιγράφουν τη λειτουργία των τριών διαθέσιμων ειδών ρομπότ.



Ειδικότερα τώρα, η κλάση Simulate (συνάρτηση της οποία εκτετείται από τη main) είναι η βασική κλάση του προγράμματος, καθώς περιέχει το GameLoop, δημιουργεί όλα τα απαραίτητα αντικείμενα (χάρτης, μενού κτλ), ελέγχει τη SplashScreen και το MainMenu και είναι υπεύθυνη για τη λειτουργία του μενού.

Η VisibleGameObject περιέχει κατά κύριο λόγο (κενές) Virtual συναρτήσεις, οι οποίες υλοποιούνται στις υποκλάσεις τις (Analyzer, Explorer, Rescue), με εξαίρεση κάποιες συναρτήσεις οι οποίες είναι κοινές και στις 3 υποκλάσεις, όπως για παράδειγμα η bool isDamaged(x,y).
Οι 3 υποκλάσεις περιέχουν τις συναρτήσεις λειτουργίας και κίνησης του κάθε ρομπότ.

Η κλάση GameObjectManager είναι μια κλάση ελέγχου όλου του προγράμματος. Περιέχει συναρτήσεις που προσθέτουν, αφαιρούν ή διαβάζουν αντικείμενα της κλάσης VisibleGameObject, καθώς επίσης και τη μέθοδο UpdateAll() η οποία τρέχει ταυτόχρονα όλες τις μεθόδους Update() κάθε VisibleGameObject.

Η κλάση Map είναι υπεύθυνη για την απεικόνιση της εικόνας του χάρτη και για την αρχικοποίηση κάθε στοιχείου του χάρτη καθώς επίσης και για το διάβασμα και τη αλλαγή τιμών αυτού.

Τέλος, οι κλάσεις MainMenu και SplashScreen είναι υπέθυνες και το αρχικό μενού και την SplashScreen αντίστοιχα.

Η κλάση Simulate και το GameLoop

Η βασικότερη κλάση του προγράμματος είναι η Simulate. Η μέθοδος Start που ανήκει σε αυτή τη κλάση είναι το μοναδικό πράγμα που εκτελείται από τη main. Η μέθοδος αυτή αρχικά δημιουργεί το κεντικό παράθυρο, δημιουργεί κάποια text που θα εμφανίζουναι στην συνέχεια πάνω στο χάρτη (pause button και lap counter) και τέλος καλεί την ShowSplashSceen και τη ShowMenu διαδοχικά.

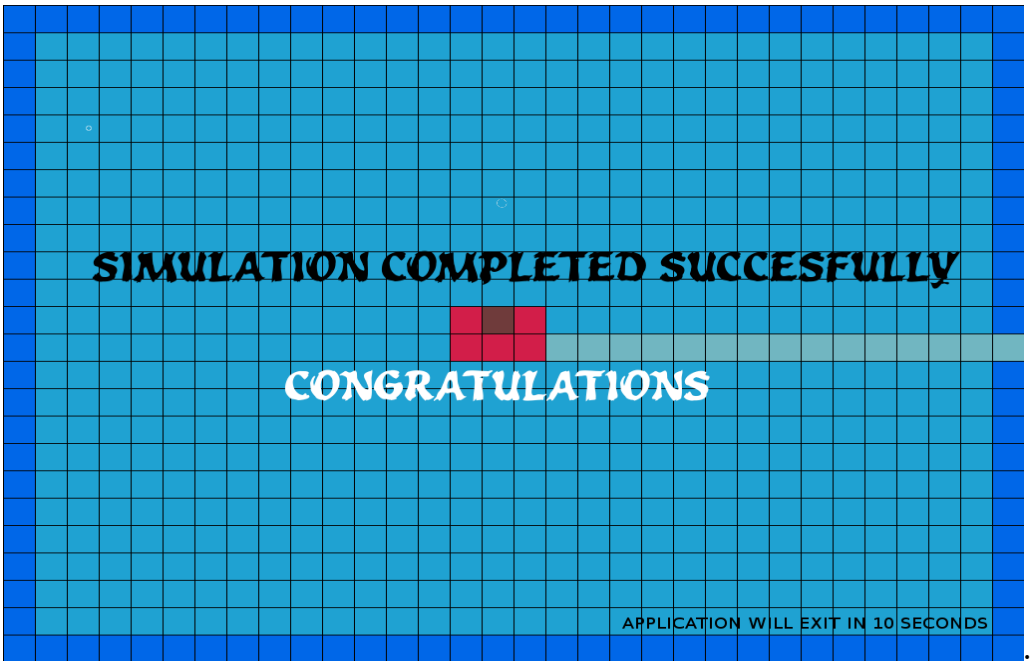
Η ShowSplashScreen δημιουργεί ένα αντικείμενο της κλάσης SplashScreen, που θα δούμε παρακάτω και εκτελεί την συνάρτηση μέλος της, Show.
Αντίστοιχη λειτουγία έχει και η ShowMenu μόνο που ανάλογα με το αποτέλεσμα που επιστρέφει έχουμε διαφορετικό αποτέλεσμα. Αν ο χρήστης επιλέξει το Exit, τότε το πρόγραμμα θα τερματιστεί με τη βοήθεια της συνάρτησης exit(0), διαφορετικά θα εκτελεστεί η **GameLoop**.

Η GameLoop τώρα εκτελεί το βασικό while loop που αφορά τον χάρτη της προσομοίωσης. Υπάρχει ένα eventLoop, το οποίο κοιτάει για το πλήκτρο Escape που επιστρέφει το πρόγραμμα στο MainMenu, για τον αριθμό 0 που τρέχει την ShowPaused και το παιχνίδι διακόπτεται και για το Close (κλείσιμο παραθύρου), που τερματίζει το πρόγραμμα.

Στην συνέχεια του Loop, καθαρίζεται κάθε φορά το παράθυρο ώστε να ενημερωθεί με τη μέθοδο clear() της sf::RenderWindow , ανανεώνονται όλα τα αντικείμενα και εμφανίζονται οι νέες τους θέσεις. Σημαντικό ρόλο εδώ παίζει η UpdateAll της GameObjectManager που ανανεώνει τις θέσεις όλων των ρομποτ.
Επιπλέον υπάρχει μια καθυστέρηση μισού δευτερολέπτου ανάμεσα σε κάθε επανάληψη με τη χρήση της εντολής του συστήματος (δουλεύει σε όλα τα λειτουργικά συστήματα), sleep 0.5 η οποία δίνεται στο σύστημα μέσω της συνάρτησης system.

```
system ("sleep 0.5");
```

Τέλος ελέγχει για το αν έχουν μαζευτεί 18 ρομπότ στη βάση οπότε και τερματίζει το παιχνίδι 10 δευτερόλεπτα αφού εμφανίσει την οθόνη επιτυχούς τερματισμού που βλέπετε παρακάτω:



Η οθόνη επιτυχούς τερματισμού

Στη κλάση Simulate βρίσκεται τέλος η μέθοδος ShowPaused, η οποία είναι υπεύθυνη για όλη τη λογική των λειτουργίων του παραθύρου παύσης (Remove, ShowInfo, add Robots κτλ) (η δημιουργία των sf::Text για το παράθυρο αυτό γίνεται στη κλάση InfoWindow).

Τα αντικείμενα map και gameobjectmanager δημιουργούνται στο .h αρχείο. Η κλάση αυτή είναι όλη static μιας και θα υπάρχει μόνο ένα στιγμιότυπό της, αν και δεν ήταν αναγκαίο.

Οθόνη έναρξης και μενού

Αφού εκκινήσουμε την εφαρμογή θα δούμε την παρακάτω εικόνα:

Οθόνης Έναρξης

Πατώντας ένα οποιοδήποτε πλήκτρο, θα εμφανιστεί το MainMenu με τις επιλογές Simulate και Exit.

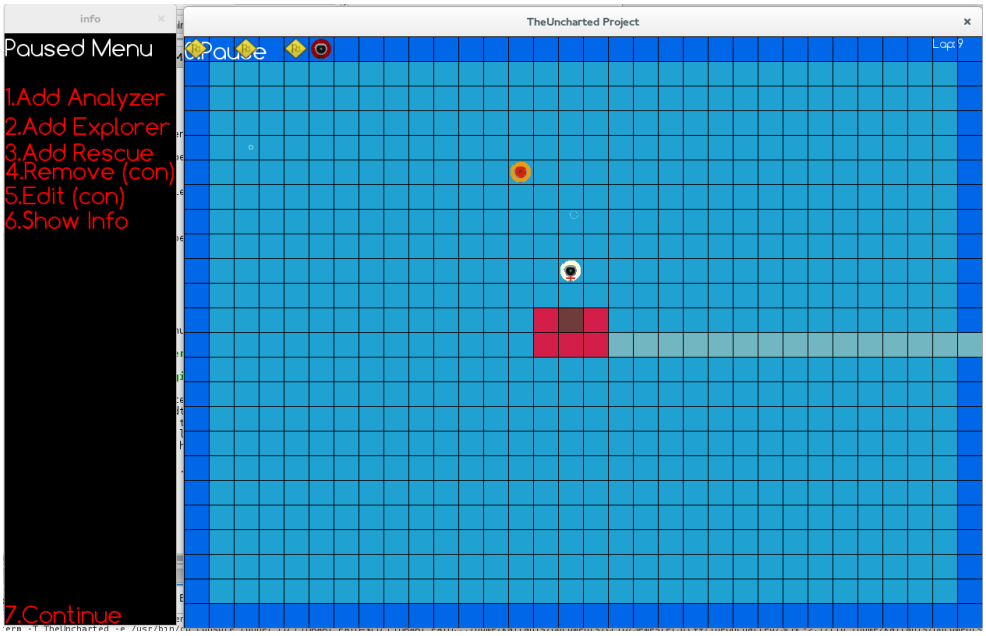
Κυρίως Μενού

Η οθόνη έναρξης έχουν υλοποιηθεί με την βοήθεια των κλάσεων SplashScreen και MainMenu αντίστοιχα.

Η κλάση **SplashScreen** έχει μόνο μία συνάρτηση-μέλος, τη Show(sf::RenderWindow & window), η οποία παίρνει σαν όρισμα ένα alias ενός RenderWindow, που το ονομάζει window.
Το μόνο που κάνει αυτή η κλάση είναι να φορτώνει την κατάλληλη εικόνα σε ένα texture και να την βάζει σε ένα sprite. Αφού το κάνει αυτό ανοίγει ένα loop το οποίο τερματίζεται, όταν πατηθεί κάποιο πλήκτρο ή κάποιο κουμπί από το ποντίκι.

Η κλάση MainMenu είναι λίγο πιο περίπλοκη. Παίρνει και αυτή στη δική της Show ένα sf::RenderWindow alias ως όρισμα για να υλοποιήσει το .draw και στο .display.
Υπάρχει αρχικά μια stuct MenuResult, η οποία περιλαμβάνει ένα sf::Rect για να κρατάει τις συντεταγμένες και ένα enum με το όνομα action για να περιγράφει την ενέργεια (exit, play ή nothing).
Η μέθοδος GetMenuResponse περιμένει από τον χρήστη να κλικάρει σε κάποια περιοχή. Όταν το κάνει αυτό καλεί την HandleClick η οποία ελέγχει αν οι συντεταγμένες που παρήγθησαν είναι μέσα σε κάποιο από τα διαστήματα play ή exit που δηλώθηκαν παραπάνω. Επιστρέφει το αποτέλεσμα στην Simulate η οποία πράτει ανάλογα με αυτό.

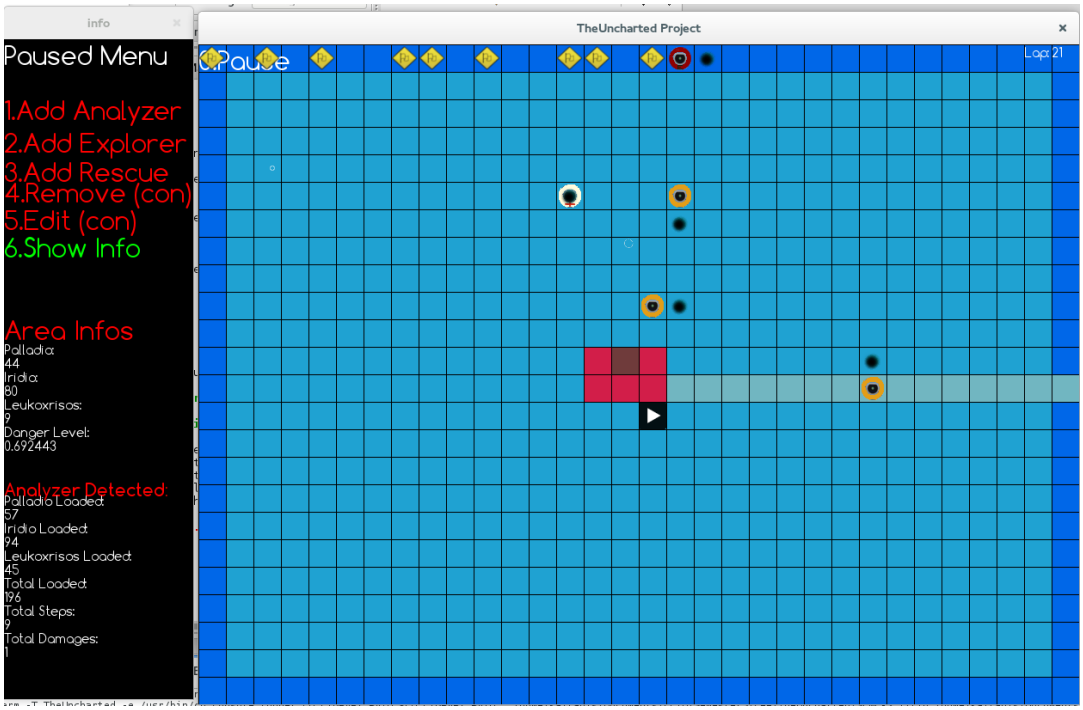
Βασικό περιβάλλον/συναρτήσεις & συμβάσεις



βασικό περιβάλλον

Το βασικό περιβάλλον αποτελείται από τον χάρτη όπου κινούνται τα ρομπότ και από το μενού παύσης, απ' όπου μπορούμε να κάνουμε όλες τις ενέργειες.

Για να ενεργοποιηθεί το μενού παύσης πατάμε το 0, έχοντας το focus στο παράθυρο του χάρτη. Για να χρησιμοποιήσουμε κάποια λειτουργία του paused menu πατάμε το κατάλληλο αριθμό έχοντας το focus εκεί.
Στις περισσότερες λειτουργίες χρειάζεται να πατήσομε το 7 για να επιστρέψουμε στην προσομοίωση εκτός από μερικές όπου χρειάζεται να πατήσουμε το play button που βρίσκεται κάτω από τη βάση.



Η κλάση map που υλοποιεί τον χάρτη δίνει τυχαίες τιμές παλλαδίου, ιριδίου και λευκόχρυσου και επικυνδυνότητας πρόσβασης σε κάθε σημείο του χάρτη (εκτός της βάσης) και περιέχει πολύ χρήσιμες συναρτήσεις που θέτουν και επιστρέφουν τιμές σε και από αυτόν.

Τα ρομπότ είναι 3 και αναγνωρίζονται από τη φωτογραφία τους.

 Ρομπότ ανάλυσης

 Ρομπότ Διάσωσης

 Ρομπότ εξερεύνησης

Όταν ένα ρομπότ είναι κατεστραμμένο σταματάει την κίνησή του και το «δαχτυλίδι» στην εικόνα του γίνεται κόκκινο όπως το ρομπότ ανάλυσης που βλέπετε παρακάτω.

Το συγκεκριμένο ρομπότ κάνει και εκσκαφή όταν η θέση του χάρτη πληρεί τις κατάλληλες προϋποθέσεις. Σε αυτή τη περίπτωση το δαχτυλίδι γίνεται πράσινο.

Συμβάσεις: Το ρομπότ εξερεύνησης χρησιμοποιεί μια διαφορετική πολιτική κίνησης από αυτή της εκφώνησης: Δε κοιτάει τις γειτονικές περιοχές, αλλά μόλις κάποιο άλλο ρομπότ πάθει βλάβη πηγαίνει κατευθείαν σε αυτό. Επίσης το ρομπότ εξερεύνησης δεν καταστρέφεται.
Επίσης το πρόγραμμα δεν τερματίζεται σε 15 γύρους αν όλα τα ρομπότ είναι κατεστραμμένα, επειδή τα ρομπότ διάσωσης θα πάνε σε αυτά αργά ή γρήγορα.
Τα ρομπότ κινούνται με ταχύτητα 1 κουτάκι / γύρο. Αυτό μπορεί να αλλάξει εύκολα από τον κώδικα (υπάρχει η μεταβλητή speed).

Εισαγωγή Ρομπότ

Αφού ενεργοποιήσουμε το PausedMenu πατώντας το 0 στο βασικό παράθυρο, πατάμε 1 για να προσθέσουμε Analyzer, 2 για Explorer, 3 για Rescue και στη συνέχεια το 7 για να μπουν τα ρομπότ στον χάρτη και να συνεχιστεί η προσομοίωση. Μπορούμε να βάλουμε ταυτόχρονα από ένα ρομπότ του κάθε είδους, όχι όμως περισσότερα.

Ο κώδικας χρησιμοποιεί την συνάρτηση Add της GameObjectManager, η οποία εισάγει στη λίστα των ρομπότ το ρομπότ που θέλουμε με τη παρακάτω γραμμή:

```
gameObjects.insert(std::pair<std::string,VisibleGameObject*>(name,gameObject));
```

Κάθε ρομπότ αποκτά ένας τυχαίο κωδικό, ο οποίος αποθηκεύεται στον πίνακα names] και χρησιμοποιείται πολύ στη συνέχεια.

Συναρτήσεις κίνησης των ρομπότ

Κάθε κλάση ρομπότ για την κίνησή της έχει μια α συνάρτηση Update (ή οποία είναι μία virtual συνάρτηση της VisibleGameObject). Εκεί γίνονται όλοι οι κατάλληλοι υπολογισμοί για την κίνηση του κάθε ρομπότ.

Ρομπότ Ανάλυσης: Το ρομπότ ανάλυσης αρχικοποιείται σε τυχαία θέση στον χάρτη με τη GenerateStartedPosition. Έπειτα με τη GenerateTarget δημιουργείται ένας τυχαίο στόχος. Μέσα στην Update το ρομπότ κινείται κατάλληλα για να φτάσει αυτόν τον στόχο. Όταν φτάσει εκεί τότε δημιουργείται ένας καινούριο κοκ.
Στην ίδια μέθοδο ελέγχεται επίσης αν το έδαφος είναι κατάλληλο για εξόρυξη με την isForMining και όταν αυτή επιστρέφει true γίνεται εξόρυξη με τη συνάρτηση Mining.

Ρομπότ Εξερεύνησης: Το ρομπότ εξερεύνησης αρχικοποιείται τυχαία σε κάποια από τις 4 γωνίες που χάρτη και ξεκινώντας από εκεί κάνει κίνηση σαν «φιδάκι», με σκοπό να καλύψει όσο το δυνατό περισσότερες επικίνδυνες περιοχές με σημαίες (Η κίνησή του είμαι μία σύμβαση). Όταν πληρούνται οι κατάλληλες προϋποθέσεις τοποθετείται μια σημαία κινδύνου στο σημείο αυτό την οποία τα ρομπότ ανάλυσης αποφεύγουν (στην ουσία δεν κάνουν εκσκαφές από εκείνα τα σημεία, οπότε δεν έχουν και κίνδυνο καταστροφής.

Ρομπότ Διάσωσης: Όπως αναφέρθηκε, τα ρομπότ αυτά πάνε κατευθείαν στο ρομπότ που έχει πρόβλημα και το επιδιορθώνουν με τη μέθοδο checkAndRepair.

Συναρτήσεις λειτουργίας των ρομπότ

Ρομπότ Ανάλυσης: Η βασική του λειτουργία είναι η εξόρυξη. Όταν λοιπόν το έδαφος είναι κατάλληλο, η isForMining επιστρέφει true. Αυτό συμβαίνει όταν κάποιο από τα 3 στοιχεία είναι μεγαλύτερο ή ίσο του 95 (με μέγιστο τα 100) ή όλα είναι μεγαλύτερα ή ίσα από 40.
Κατά τη διαδικασία της εξόρυξης τώρα (συνάρτηση Mining) , το ρομπότ ελέγχει αν αυτά που πρόκειται να πάρει είναι <= από το MaxLoad (το οποίο είναι ίσο με 2000). Αν είναι τα παίρνει όλα, αλλιώς παίρνει ακριβώς όσα χρειάζεται για να ολοκληρώσει το φορτίο του.

Ρομπότ Εξερεύνησης: Στην Update, όταν η isForFlag επιστρέψει true, τότε τοποθετείται σημαία κινδύνου στο κατάλληλο σημείο. Η isForFlag επιστρέφει true όταν η επικινδυνότητα είναι μεγαλύτερη του 0.6.
Υπάρχει ένας πίνακας που χωράει 768 sf::Sprite (δεν έγινε δυναμική υλοποίηση εδώ). Κάθε φορά που είναι να εμφανιστεί μια νέα σημαία μπαίνει στον πίνακα και όλες εμφανίζονται μαζί σε ένα for loop.

Ρομπότ Διάσωσης: Η checkAndRepair είναι υπεύθυνη για τη βασική λειτουργία του ρομπότ διάσωσης. Κοιτάει σε κάθε βήμα της κίνησής του αν χρειάζεται βοήθεια κάτι στο σημείο που βρίσκεται (υπάρχει ο πίνακας NeedHelp για το λόγο αυτό). Αν υπάρχει αποθηκεύει σε ένα προσωρινό VisibleGameObject το ρομπότ αυτό (με τη βοήθεια της Get από την GameObjectManager) και εκτελεί τη setDamagedFalse(), μια virtual

συνάρτηση της VisibleGameObject η οποία κάνει τις τελευταίες προφανείς ενέργειες.

Επεξεργασία στοιχείων του χάρτη & bash script

Οι λειτουργίες του PausedMenu υλοποιούνται στη μέθοδο ShowPaused της Simulate, εκτός από το γραφικό περιβάλλον που υλοποιείται στη InfoWindow.

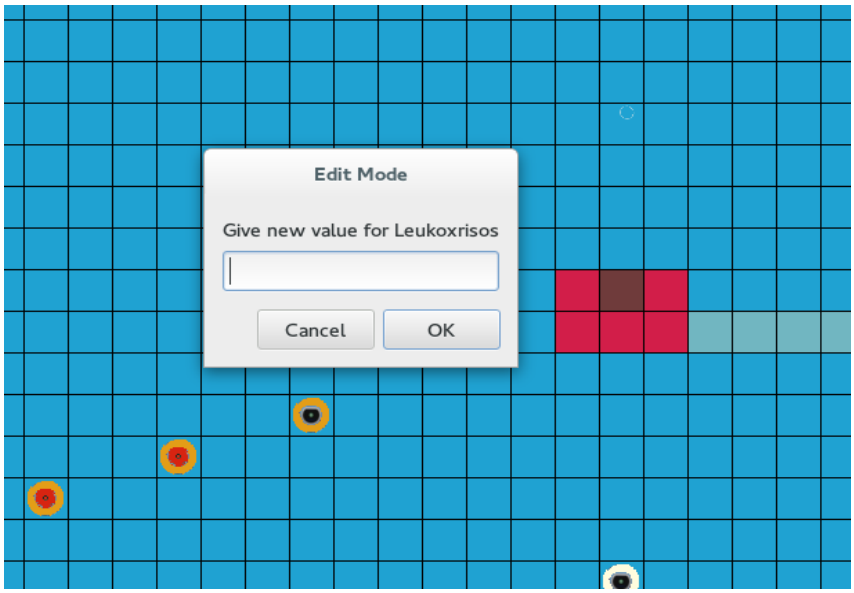
Η επεξεργασία των στοιχείων του χάρτη υλοποιήθηκε αρχικά με κονσόλα, η οποία όμως δεν άνοιγε όταν εκκινούσα το πρόγραμμα εκτός code::blocks. Οπότε χρησιμοποιήθηκε ένα απλό bash script γι αυτό το σκοπό, το list.sh.

Το script αυτό χρησιμοποιεί το zenity (είναι προεγκατεστημένο σχεδόν σε όλες τις διανομές Linux, το οποίο εμφανίζει ένα απλό γραφικό περιβάλλον για το σκοπό αυτό. Ανάλογα με με την απάντηση του χρήστη για το ποιο στοιχείο θέλει να αλλάξει από το κελί που επέλεξε, γράφεται στο αρχείο RE.txt η κατάλληλη απάντηση.
Έπειτα ανοίγει ένα παράθυρο εισαγωγής κειμένου και η απάντησε αποθηκεύεται στο ENTRY.txt.
Στη συνέχεια η C++ διαβάζει αυτό τα αρχεία και πράττει ανάλογα, χρησιμοποιώντας την setPalladio κτλ για να αλλάξει τις τιμές.

Ακόλουθεί ο κώδικας του bash script:

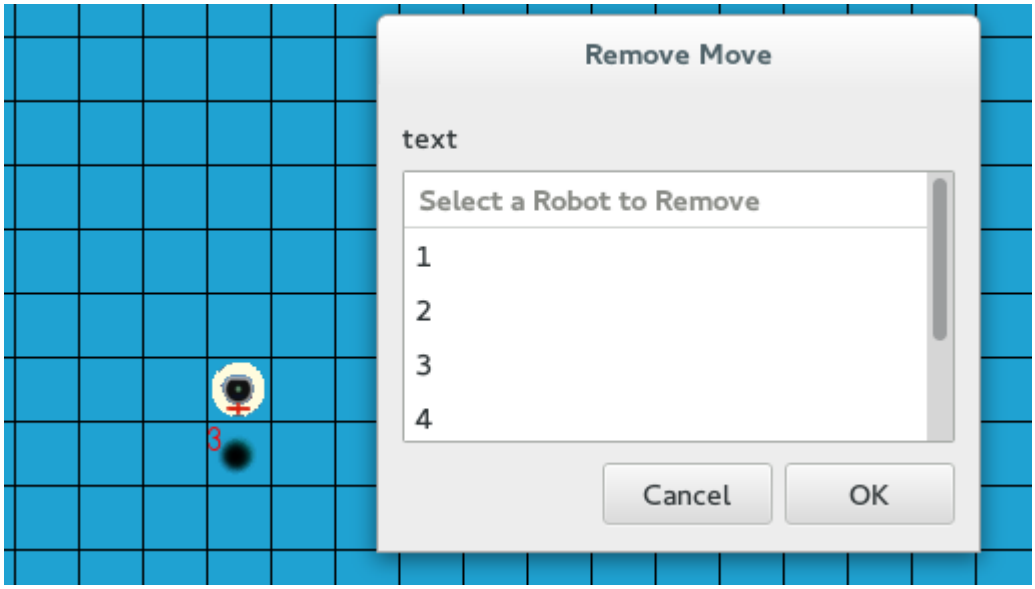
```
#!/bin/bash

RE=`zenity --list --column="What element would toy like to change?" Leukoxrisos
Iridio Palladio Continue --title="Entry Mode" --width=600 --height=200`
if [ $? = 1 ]; then
        exit
fi
if [ $RE = "Leukoxrisos|Leukoxrisos" ]; then
        ENTRY=`zenity --entry --text="Give new value for Leukoxrisos" --title="Edit
Mode"`
elif [ $RE = "Iridio|Iridio" ]; then
        ENTRY=`zenity --entry --text="Give new value for Iridio" --title="Edit Mode"`
elif [ $RE = "Palladio|Palladio" ]; then
        ENTRY=`zenity --entry --text="Give new value for Palladio" --title="Edit Mode"`
elif [ $RE = "Continue|Continue" ]; then
        echo $RE > RE.txt
        exit
fi
echo $RE > RE.txt
echo $ENTRY > ENTRY.txt
```

Αυτή η λειτουργία είναι διαθέσιμη μόνο σε διανομές Linux, εξαιτίας του script.



Αφαίρεση ρομπότ & bash script

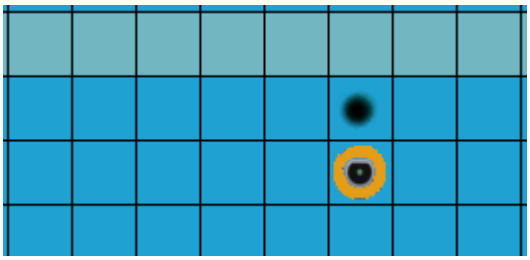
Η λειτουργία αυτή χρησιμοποιεί έναν παρόμοιο μηχανισμό, εξαιτίας κάποιων bugs (κανονικά έπρεπε να φεύγουν με το κλικ σε κάθε ρομπότ).
Τώρα, εμφανίζεται sf:Text με έναν αριθμό (ο οποίος αλλάζει δυναμικά), και ο χρήστης επιλέγει ποιο θέλει να απομακρύνει από την προσομοίωση.



Από τη μεριά της C++ χρησιμοποιείται η remove της GameObjectManager για τη διαγραφή.

Εμφάνιση στοιχείων του χάρτη

Όταν ο χρήστης πατάει το 6 στο μενού παύσης τότε είμαστε στο Show Info Mode. Το πρόβλημα εδώ ήταν ένα περίεργο bug όπου οι συντεταγμένες που κλίκαρε ο χρήστης διέφεραν κατά ένα τετραγωγάκι από αυτές του Sprite. Αυτό «επιλύθηκε» με τις μαύρες κουκίδες. Δεν κάνουμε κλικ στο ρομπότ για να δούμε τα στοιχεία του, αλλά στη μαύρη κουκίδα δίπλα του. Εγώ το αποδέχτηκα :)



Αν κάνουμε κλικ σε ένα «άκυρο» σημείο του πίνακα θα δούμε τα στοιχεία για αυτό. Αν κάνουμε σε ένα ρομπότ θα δούμε και τα στοιχεία του κελιού του χάρτη, αλλά και του ρομπότ.

Όλο αυτό υλοποιείται στην InfoWindows όπου κάθε sf::Text παίρνει τιμές από τις μεθόδους των στιγμιοτύπων που συμμετέχουν στην προσομοίωση και τις εμφανίζει με τον κατάλληλο τρόπο.

Αν τώρα κάνουμε κλικ στο καφέ κουτάκι της βάσης θα δούμε τα στοιχεία της βάσης. Να σημειωθεί ότι τα ρομπότ ανάλυσης όταν φτάσουν στο μέγιστο φορτίο τοποθετούνται στα 3 κουτάκια της βάσης και στα υπόλοιπα 15 που είναι με γαλάζιο χρώμα (στα 18 τελειώνει η προσομοίωση).

Παράμετροι εκτέλεσης του προγράμματος

Στο Fedora χρειάζεται να εγκαταστήσετε το libGLEW 1.5 (αφού πρώτα σβήστε την έκδοση που πιθανόν έχετε).
Αυτό γίνεται δίνοντας:

$ sudo yum remove libGLEW

και μετά επαναστατώντας το libGLEW που βρίσκεται στον φάκελο libs του προγράμματος.

Τέλος πρέπει να ανοίξετε το /etc/ld.so.conf ως root με κάποιο κειμενογράφο πχ με το nano

$ sudo nano /etc/ld.so.conf

και να γράψετε σε νέα γραμμή το ολοκληρωμένο path που οδηγεί στο SFML-2.1/lib του προγράμματος. Τέλος δίνετε

$ sudo ldconfig

*Στον φάκελλο του προγράμματος έχω φτιάξει ένα σκριπτάκι (**fedora_setup.sh**) που κάνει αυτόματα όλη αυτή του δουλειά (καλύτερα να το τρέξετε από γραμμή εντολών).

Το πρόγραμμα θα δουλέψει κανονικά από code::blocks. Αν το κάντε χειροκίνητα μπορείτε να αντιγράψετε το εκτελέσιμο που υπάρχει στο bin/Debug στον φάκελο με τα άλλα αρχεία και θα τρέξει. Το fedora_setup θα δημιουργήσει το εκτελέσιμο με compile. Για να κάνετε εσείς το compile αρκεί να δώσετε:

$ g++ -c *.cpp -ISFML-2.1/include
$ g++ *.o -o TheUncharted -LSFML-2.1/lib -lsfml-graphics -lsfml-window -lsfml-system


ubuntu

Στο ubuntu χρειάζεται η ίδια διαδικασία με τη διαφορά ότι απαιτείται η libGLEW 1.7 που μπορείτε να βρείτε στο φάκελο libs του project.


Windows (δεν έχει δοκιμαστεί)

Σύμφωνα με το επίσημο sait της sfml εξαρτήσεις όπως η libGLEW δεν χρειάζονται στα windows. Απλώς κάνετε compile χρησιμοποιώντας τις βιβλιοθήκες για τα windows από εδώ: http://www.sfml-dev.org/download/sfml/2.1/
Στο λειτουργικό αυτό ΔΕΝ δουλεύουν οι λειτουργίες Remove και Edit, επειδή είναι γραμμένες σε bash.

Βιβλιογραφία

C++ Προγραμματισμός PJ DEITL HM. DEITEL
TheNewBoston.org
SFML tutorials & documentation

**Σημαντική βοήθεια ήταν το Game for Scratch C++ and SFML Edition
http://www.gamefromscratch.com/page/Game-From-Scratch-CPP-Edition.aspx
από το οποίο «δανείστηκα» τη δομή του προγράμματος, τη λογική για την υλοποίηση του project και μερικά έτοιμα κομμάτια κώδικα.