

Tower Defense

Ιωάννης Ευθυμίου

A.M. 1072680

Έτος 4^ο

Τομέας: Υπολογιστές Λογισμικό και Υλικό

Μάθημα: Γραφικά και Εικονική Πραγματικότητα



Περίληψη

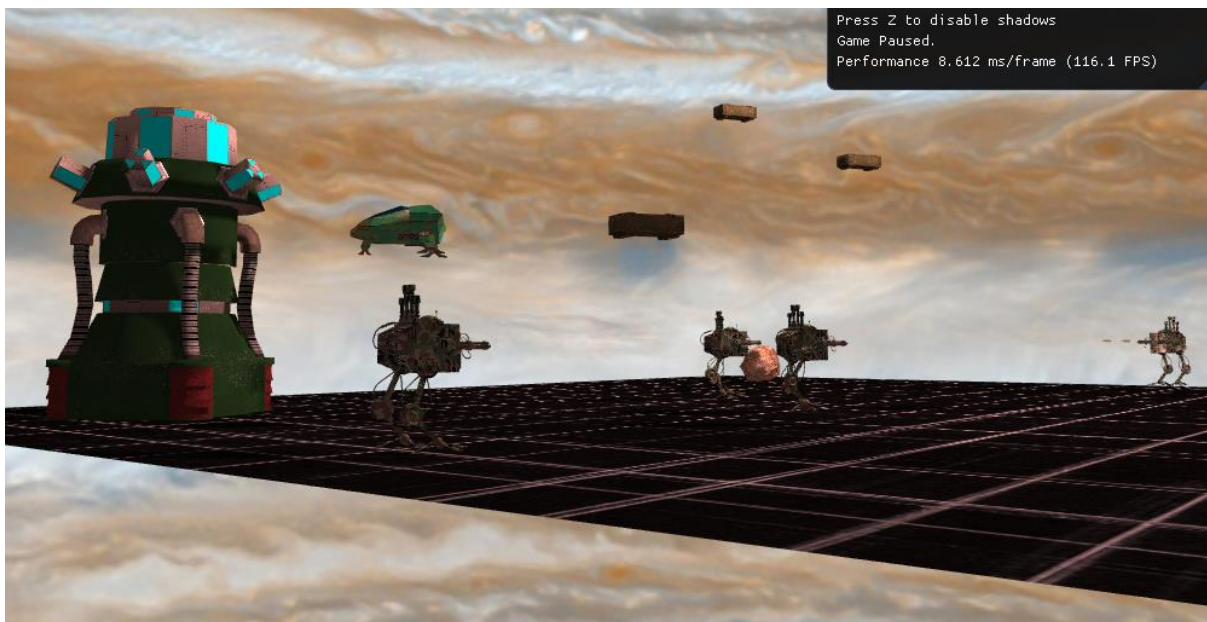
Η παρακάτω αναφορά πραγματεύεται την παρουσίαση ενός παιχνιδιού τύπου Tower Defense, που υλοποιήθηκε στα πλαίσια του μαθήματος Γραφικά και Εικονική Πραγματικότητα. Πρώτα γίνεται μια επεξήγηση του παιχνιδιού με εμβάθυνση στον τρόπο που παίζεται και στους χαρακτήρες του. Έπειτα εξηγούνται οι αλγοριθμικές τεχνικές που χρησιμοποιήθηκαν στη δημιουργία του, όπως το μοντέλο Phong για την φωτοσκίαση. Η έκθεση εμπλουτίζεται με στιγμιότυπα από το παιχνίδι.

Εισαγωγή στο Παιχνίδι

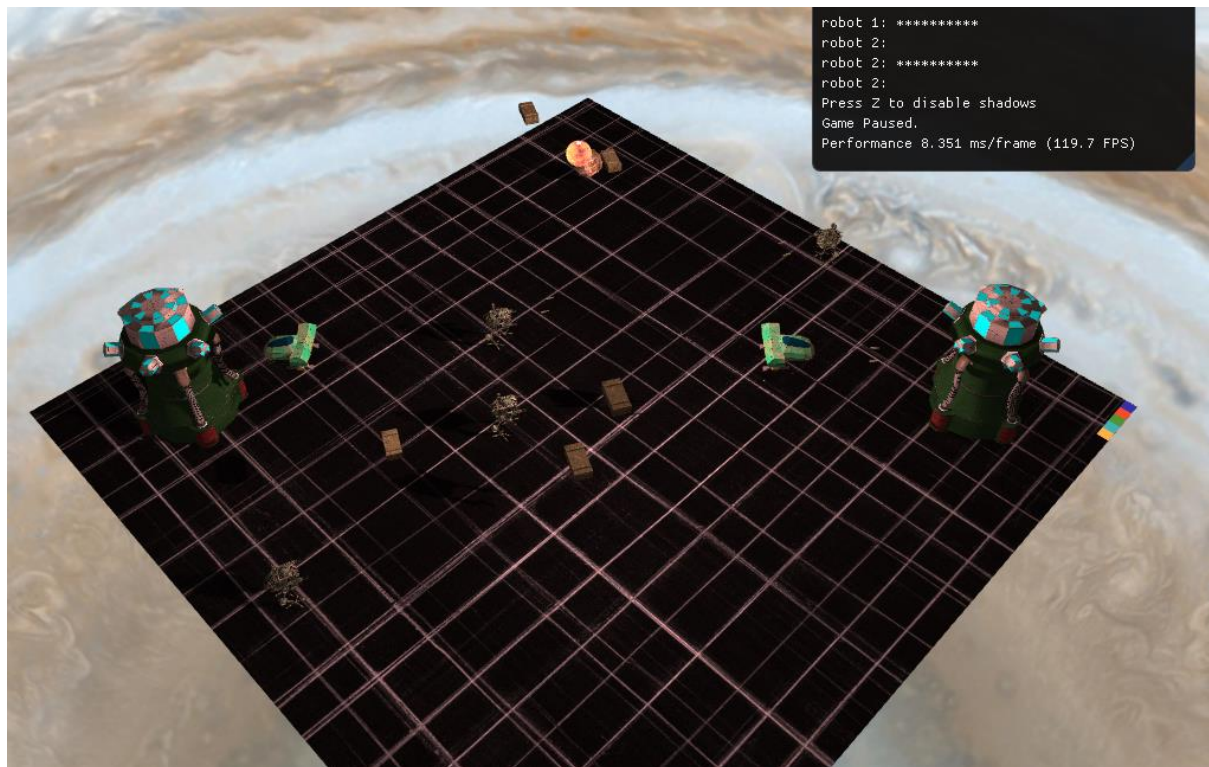
Το παιχνίδι που αναπτύχθηκε είναι τύπου Tower Defense. Στο παιχνίδι αυτό υπάρχουν δύο παίκτες οι οποίοι λειτουργούν αυτόματα, δηλαδή ο χρήστης δεν μπορεί να επηρεάσει το αποτέλεσμα του παιχνιδιού, ωστόσο και δυο παίκτες έχουν την ίδια πιθανότητα να κερδίσουν. Κάθε παίκτης έχει μια βάση (πύργος) που καλείται να υπερασπιστεί, καθώς και να βρει τρόπο να επιτεθεί στην αντίπαλη βάση. Οι παίκτες λοιπόν έχουν στη διάθεση τους δύο χαρακτήρες. Ο πρώτος χαρακτήρας μπορεί να κάνει μόνο επίθεση, και ο δεύτερος που μπορεί να κάνει επίθεση αλλά μπορεί και να αμυνθεί σε τυχόν επιθέσεις του αντιπάλου. Ο πύργος του κάθε παίκτη στην αρχή του παιχνιδιού έχει το μέγιστο σε ζωή. Όταν γίνονται επιθέσεις οι χαρακτήρες έχουν την δυνατότητα να μειώσουν την ζωή του αντίπαλου πύργου. Ο πρώτος παίκτης που καταφέρει να μηδενίσει την ζωή του αντίπαλου πύργου είναι ο νικητής του παιχνιδιού.

Περιβάλλον Παιχνιδιού

Το παιχνίδι διαδραματίζεται στο εσωτερικό μιας σφαίρας η οποία έχει texture την επιφάνεια του πλανήτη Δία, και όλα τα models του παιχνιδιού βρίσκονται πάνω από ένα επίπεδο.



Σχ.1 Περιβάλλον Παιχνιδιού 1



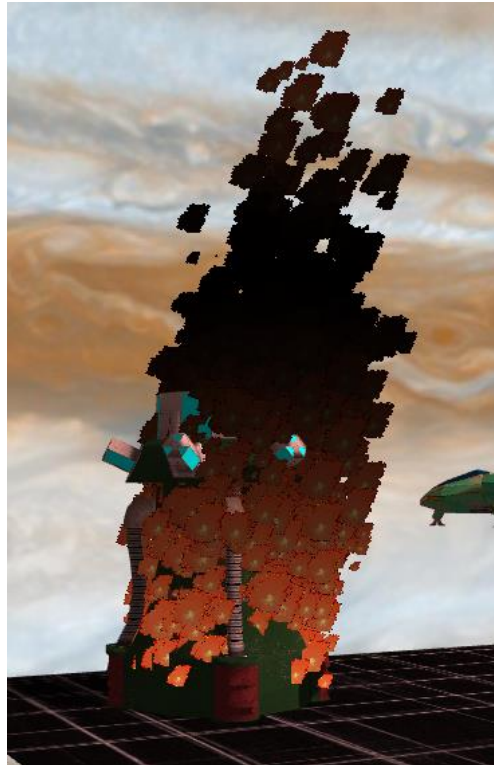
Σχ.2 Περιβάλλον Παιχνιδιού (Κάτοψη)

Πύργος

Κάθε παίκτης έχει έναν πύργο. Η ζωή του πύργου αντιστοιχεί στην ζωή του παίκτη, δηλαδή όταν κάποιος παίκτης καταφέρει να μηδενίσει την ζωή του αντίπαλου πύργου τότε κερδίζει και το παιχνίδι. Στο τέλος του παιχνιδιού ο χαμένος πύργος φλέγεται.



Σχ. 3 Πύργος

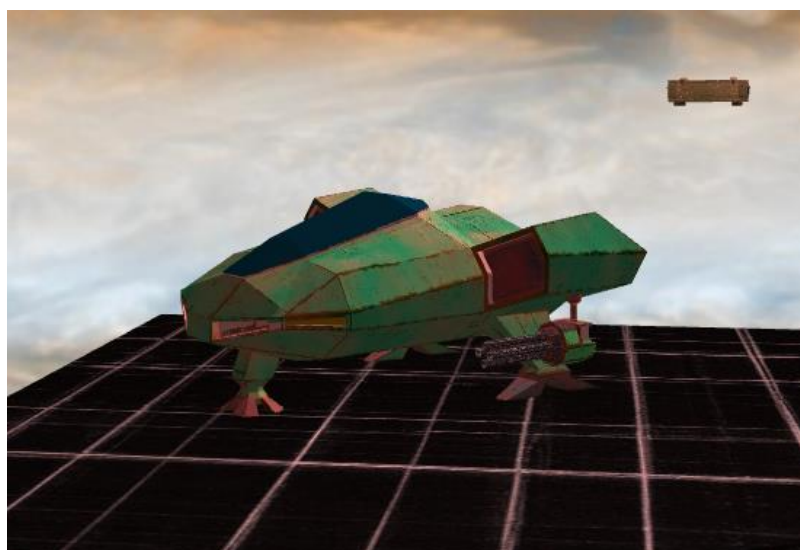


Σχ.4 Πύργος που φλέγεται

Χαρακτήρες Παιχνιδιού

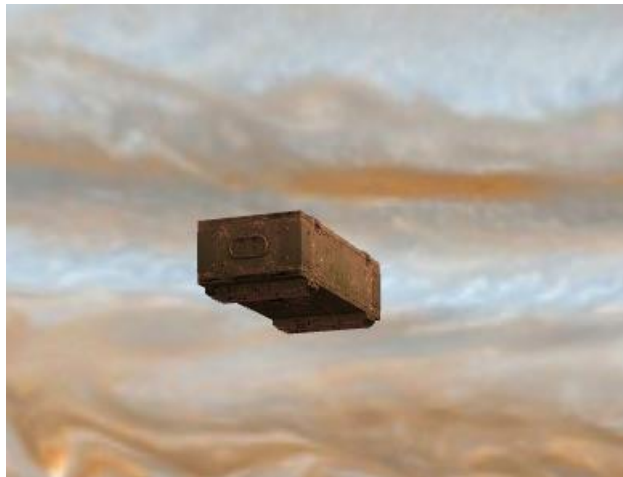
Aircraft

Ο χαρακτήρας aircraft είναι όπως λέει και το όνομα του είναι ένα ιπτάμενο αεροπλανάκι. Κάθε παίκτης έχει στην διάθεση του μόνο ένα. Το aircraft έχει την δυνατότητα να κάνει μόνο επίθεση στον αντίπαλο πύργο. Το κάθε aircraft έχει ένα απόθεμα σε σφαίρες (ammo). Όταν έχει διαθέσιμες σφαίρες πηγαίνει προς τον αντίπαλο πύργο και πυροβολεί. Όταν πυροβολεί μειώνεται η ζωή του αντίπαλου πύργου. Η μείωση της ζωής είναι αντιστρόφως ανάλογη της απόστασης του πύργου με το aircraft, δηλαδή όσο πιο κοντά είναι στον πύργο τόσο περισσότερο μειώνεται η ζωή του πύργου.

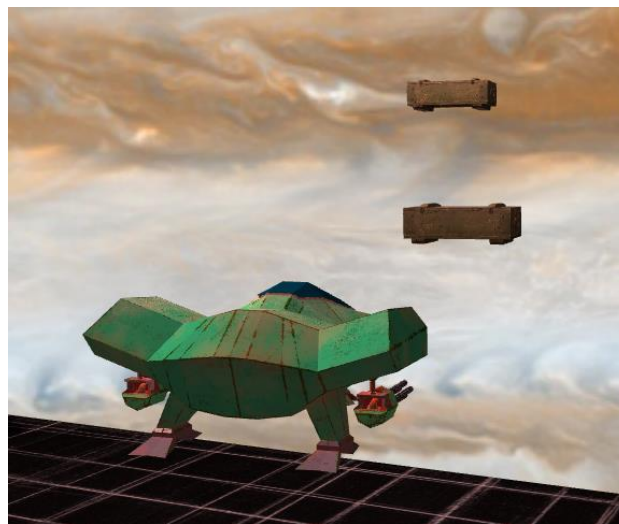


Σχ.5 Aircraft

Στην περίπτωση που οι σφαίρες τελειώσουν τότε το aircraft καλείται να αναζητήσει σφαίρες. Για τον λόγο αυτό υπάρχουν στο χώρο σε τυχαίες θέσεις ammo packages, δηλαδή κουτιά που περιέχουν σφαίρες για τα aircrafts. Όταν λοιπόν τελειώσουν οι διαθέσιμες σφαίρες, το aircraft αναζητεί το κοντινότερο package και πηγαίνει να το δεσμεύσει. Όταν φτάσει στην θέση του package τότε γεμίζει το απόθεμα του σε σφαίρες και το package αναγεννιέται σε μια νέα τυχαία θέση του χώρου, και το aircraft πηγαίνει προς τον αντίπαλο πύργου για να κάνει επίθεση. Σημειώνεται ότι κάθε χρονική στιγμή υπάρχουν 5 ammo packages σε τυχαίες θέσεις από τα οποία τα aircrafts μπορούν να ανεφοδιαστούν.



Σχ. 6 Package ammo



Σχ. 7 Το aircraft κατευθύνεται προς το ammo package

Robot

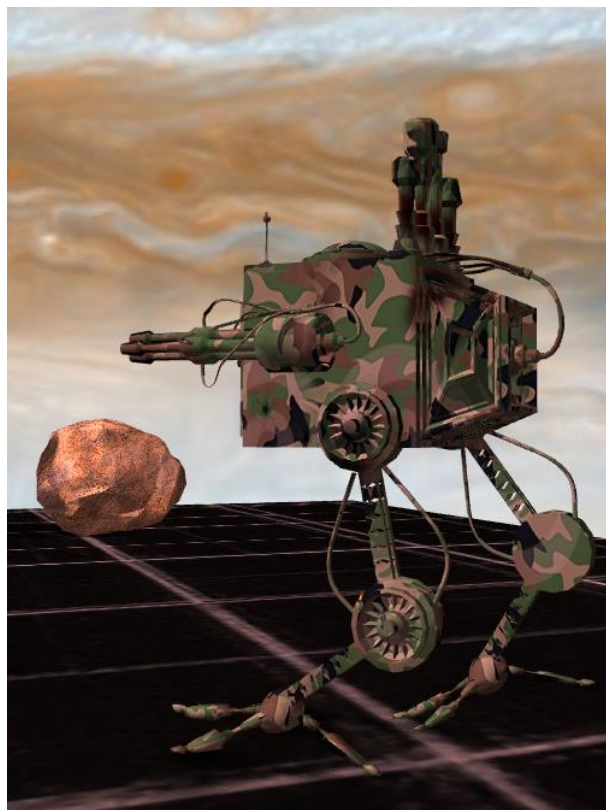
Το robot είναι ένας ακόμα χαρακτήρας του παιχνιδιού ο οποίος μπορεί να χρησιμοποιηθεί είτε για επίθεση είτε για άμυνα, προστατεύοντας τον πύργο του παίκτη. Κάθε παίκτης έχει στη διάθεση του τρία το πολύ robots, ανάλογα την φάση στην οποία βρίσκεται το παιχνίδι (θα αναλυθούν παρακάτω οι φάσεις). Το robot μπορεί να έχει διαφορετικό στόχο ανάλογα με το εάν αμύνεται ή επιτίθεται. Έχει την δυνατότητα να πυροβολήσει μειώνοντας την ζωή του στόχου του. Η μείωση αυτή είναι αντιστρόφως ανάλογη της απόστασης του robot-στόχου όπως ακριβώς και στην περίπτωση των aircrafts.

Σημειώνεται ότι τα robot έχουν απεριόριστες σφαίρες και δεν απαιτείται να εφοδιαστούν. Ακόμη κάθε robot έχει ζωή η οποία μπορεί να μειωθεί όταν δέχεται επίθεση.

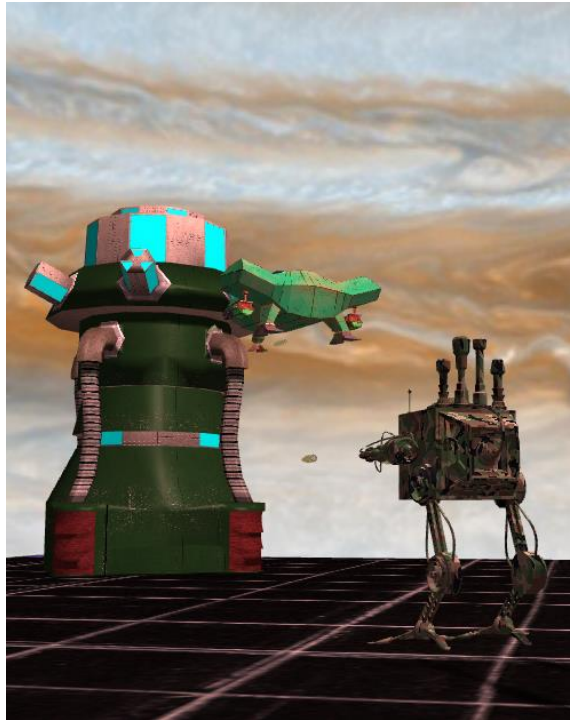
Κάθε robot λοιπόν εμφανίζεται αρχικά σε μια θέση κοντά στο πύργο του κάθε παίκτη. Το robot αρχικά αναζητεί αντίπαλο robot. Σε περίπτωση που ο αντίπαλος δεν έχει διαθέσιμα robot, τότε ο στόχος του είναι ο αντίπαλος πύργος. Στην περίπτωση αυτή κατευθύνεται προς τον στόχο του και πυροβολεί μειώνοντας έτσι την ζωή του αντίπαλου πύργου.

Στην περίπτωση που ο αντίπαλος έχει (εν ζωή) robot που προσπαθούν να επιτεθούν, τότε τα robot μάχονται μεταξύ τους με σκοπό να προστατέψουν τον πύργο τους. Κάθε robot ψάχνει το κοντινότερο αντίπαλο robot για να επιτεθεί.

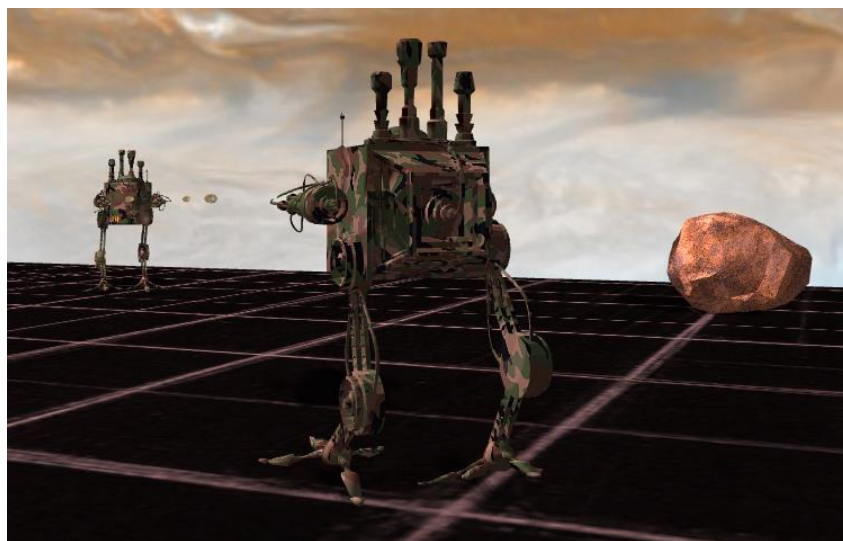
Όταν η ζωή ενός robot εξουδετερωθεί τότε μπαίνει στην φάση της «αναγέννησης». Πιο συγκεκριμένα σε κάθε frame το robot από την στιγμή που πεθαίνει έχει πιθανότητα 0.125% να αναγεννηθεί. Στην περίπτωση που αναγεννιέται επανατοποθετείται στο παιχνίδι στην θέση κοντά στον πύργο του και καλείται να επιτεθεί ή να αμυνθεί ανάλογα με την κατάσταση το αντιπάλου όπως εξηγήθηκε παραπάνω.



Σχ.8 robot



Σχ.9 robot επιτίθεται στον αντίπαλο πύργο



Σχ. 10 robot αμύνεται, και μάχεται με άλλο robot

Φάσεις του παιχνιδιού

Υπάρχουν τρεις φάσεις του παιχνιδιού, που καθορίζουν και τα κύματα επίθεσης. Αναλυτικότερα στην πρώτη φάση οι πύργοι έχουν διαθέσιμα μόνο τα aircrafts, που κάνουν επίθεση. Όταν η ζωή ενός από τους δύο πύργους πέσει κάτω από το 80% τότε περνάμε στην δεύτερη φάση του παιχνιδιού, που ο κάθε παίκτης αποκτά επιπλέον δύο robots. Τέλος, όταν η ζωή ενός από τους δύο πύργους πέσει κάτω από το 50%, τότε περνάμε στην τρίτη φάση του παιχνιδιού που ο κάθε παίκτης έχει διαθέσιμο ένα επιπλέον robot.



Σχ.11 Φάση 1 (1 aircraft για κάθε παίκτη)



Σχ.12 Φάση 2 (1 aircraft και 2 robot για κάθε παίκτη)



Σχ.13 Φάση 3 (1 aircraft και 3 robots για κάθε παίκτη)

Υλοποίηση του παιχνιδιού

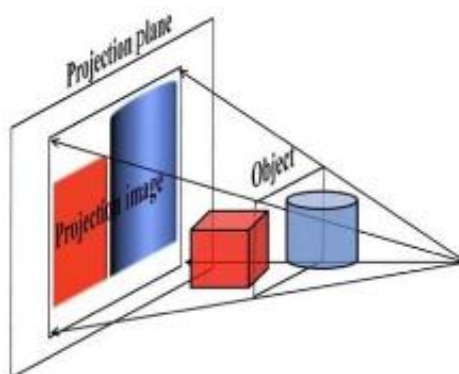
Για την υλοποίηση του παιχνιδιού έγινε χρήση της γλώσσας προγραμματισμού C++ καθώς και της βιβλιοθήκης OpenGL.

Για την υλοποίηση του παιχνιδιού έχουν αναπτυχθεί οι παρακάτω κλάσεις.

- Aircraft
- Animation
- BulletEmitter
- Camera
- Moving_obj
- FireEmitter
- IntParticleEmitter
- Drawable
- Moving_object
- RigidBody
- Robot
- Light

Πρώτα η **RigidBody** είναι μια κλάση η οποία χρησιμοποιήθηκε στην περίπτωση που εφαρμόζεται μια δύναμη πάνω σε ένα model. Πιο συγκεκριμένα η κλάση αυτή υλοποιεί την μέθοδο Runge-Kutta η οποία είναι μια επαναληπτική μέθοδος που χρησιμοποιείται για τον υπολογισμό της θέσης του model ανάλογα με την επιτάχυνση του.

Camera, η κλάση αυτή χρησιμοποιείται για την μετακίνηση της κάμερας στον χώρο με την χρήση του πληκτρολογίου καθώς και του ποντικιού από τον χρήστη. Ακόμη εκεί δηλώνεται ο τρόπος με τον οποίο η κάμερα αντιλαμβάνεται τα αντικείμενα του χώρου. Υπάρχουν δύο επιλογές orthographic ή perspective. Στην εργασία αυτή έχει χρησιμοποιηθεί perspective, δηλαδή τα αντικείμενα τα οποία βρίσκονται μακριά είναι πιο μικρά σε σύγκριση με αυτά που βρίσκονται πιο κοντά σε σχέση με την θέση της κάμερας.



Σχ. 14 Perspective

Σημειώνεται ότι ο χρήστης μπορεί να μετακινήσει την κάμερα με τα πλήκτρα:

- q,e για μετακίνηση πάνω κάτω
- w,s για μετακίνηση μπρος πίσω
- a,d για μετακίνηση αριστερά δεξιά

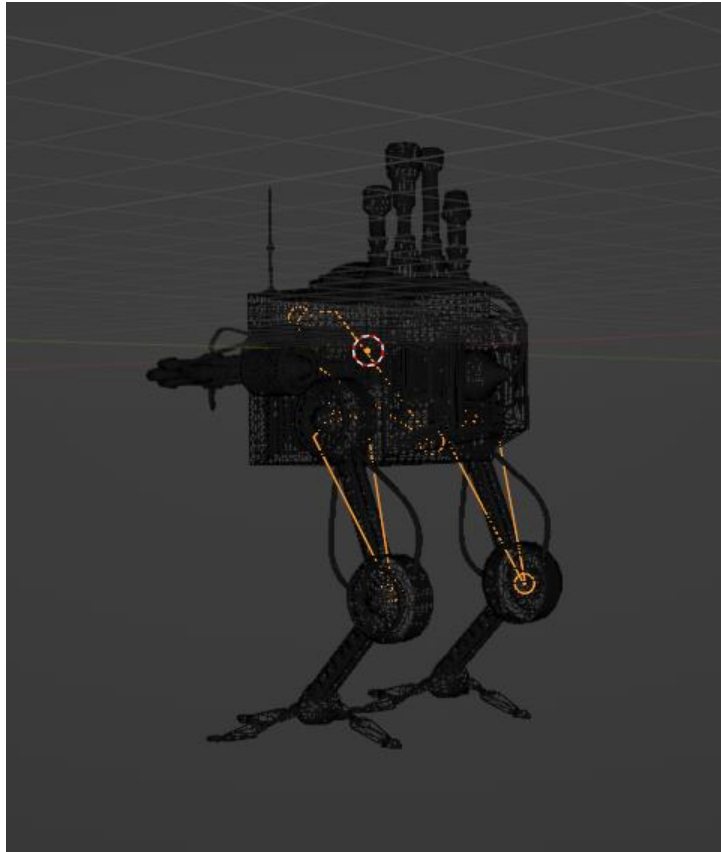
Drawable, η κλάση αυτή χρησιμοποιείται για να φορτώσει τα models, δηλαδή τα obj αρχεία.

Moving_Obj, η κλάση αυτή κληρονομεί την κλάση RigidBody. Οι επιπλέον λειτουργίες που έχει αφορούν την κίνηση ενός model. Πιο συγκεκριμένα κάθε αντικείμενο moving_obj έχει ένα στόχο. Ο στόχος αυτός είναι ένα σημείο στον χώρο, που το αντικείμενο θέλει να φτάσει. Επομένως, το αντικείμενο που έχει μια αρχική ταχύτητα αλλάζει την θέση του μέσω της αλλαγής της ταχύτητας του και της επιτάχυνσης. Επειδή θεωρητικά θα μπορούσε το αντικείμενο να φτάσει απευθείας στο target (πολύ μεγάλη επιτάχυνση/ταχύτητα), υπάρχει μια μέγιστη ταχύτητα και μια μέγιστη επιτάχυνση που το αντικείμενο δεν μπορεί να ξεπεράσει. Επομένως, το αντικείμενο φτάνει σταδιακά στο target. Ακόμη, όταν το αντικείμενο πλησιάζει τον στόχο του, τότε η μέγιστη ταχύτητα και επιτάχυνση μειώνεται ανάλογα με την απόσταση θέση αντικειμένου-target, ούτως ώστε να υπάρχει μια επιβράδυνση του αντικειμένου.

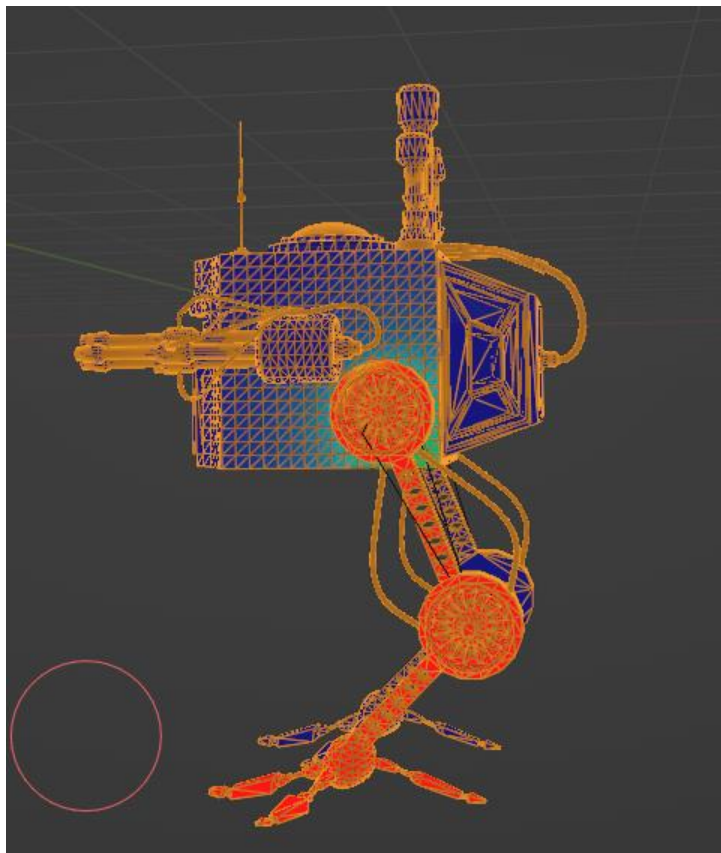
Aircraft, η κλάση αυτή κληρονομεί την κλάση Moving_obj και την Drawable. Χρησιμοποιείται για την δημιουργία των χαρακτήρων aircraft. Οι επιπλέον λειτουργίες που έχει σε σχέση με τις κλάσεις που κληρονομεί είναι ο ανεφοδιασμός σε ammo των aircrafts, η αναγέννηση ammo packages, η διαχείριση του target (της Moving_obj) και η διαχείριση της ζωής του αντίπαλου πύργου όταν το aircraft επιτίθεται. Πιο αναλυτικά, εάν οι σφαίρες τελειώσουν, τότε υπάρχει συνάρτηση που υπολογίζει ποιο είναι το πιο κοντινό ammo package και αλλάζει το target της Moving_obj, επομένως το aircraft μετακινείται προς το πλησιέστερο ammo package για ανεφοδιασμό. Τέλος, σε περίπτωση που ένα aircraft βρίσκεται κοντά στον αντίπαλο πύργο τότε εκτελεί επίθεση και η ζωή του αντίπαλου πύργου μειώνεται.

Animation, η κλάση αυτή φορτώνει τα αρχεία τύπου dae, τα οποία περιλαμβάνουν animation των models. Πιο αναλυτικά, για το animation του robot ακολουθήθηκαν τα παρακάτω βήματα.

Πρώτα επιλέχθηκε ένα obj file από το διαδίκτυο. Στο αρχείο αυτό μέσω blender χωρίστηκαν γεωμετρικές οντότητες (bones) οι οποίες συνδέονται με ιεραρχικό τρόπο και σχηματίζουν ένα γράφο σκηνής. Τα bones συνδέονται σε μια αλυσίδα ελέγχου για να εξαρτάται η κίνηση ενός ή περισσότερων γεωμετρικών στοιχείων από την κίνηση μιας οντότητας γονέα, δημιουργώντας μια κινηματική αλυσίδα. Στο παράδειγμα του robot, υπάρχουν τρία bones, τα δύο αφορούν τα «πόδια» του robot ενώ το τρίτο αφορά το υπόλοιπο model που θεωρείται κύρια γεωμετρική οντότητα. Τα πόδια είναι απόγονοι, επομένως η κίνηση τους γίνεται με βάση το σύστημα συντεταγμένων του κύριου bone (ρίζα). Είναι σημαντικό να σημειωθεί ότι κάθε vertex μπορεί να συσχετιστεί με κανένα ή πολλά bones. Έτσι όταν κινείται το bone να κινείται και κάποιο σημείο. Πόσο όμως θα κινείται ένα σημείο όταν κινείται ένα bone; Η απάντηση είναι με το skinning, δηλαδή της ανάθεσης βαρών ανά bone σε κάθε vertex. Για παράδειγμα, τα σημεία που αποτελούν τα πόδια του robot έχουν μέγιστο βάρος από τα bones «πόδια». Ενώ τα υπόλοιπα σημεία του model έχουν μέγιστο βάρος από το κύριο bone.



Σχ.15 bones in model



Σχ. 16 weighting

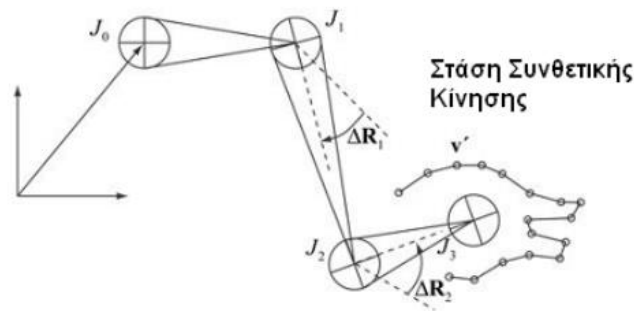
Έστω ότι θέλουμε να μετακινήσουμε το ένα πόδι του robot. Ουσιαστικά θα εκτελέσουμε έναν μετασχηματισμό για όλα τα vertices που έχουν μη μηδενικό βάρος από το bone του ποδιού. Επειδή το bone του ποδιού είναι απόγονος του κύριου bone. Θα πρέπει να γίνουν οι εξής μετασχηματισμοί.

1. Μεταφορά στο τοπικό σύστημα συντεταγμένων της ρίζας,
2. Μεταφορά στο σύστημα συντεταγμένων κόσμου

Και ύστερα να εφαρμοστεί ο ζητούμενος μετασχηματισμός στο πόδι του robot.

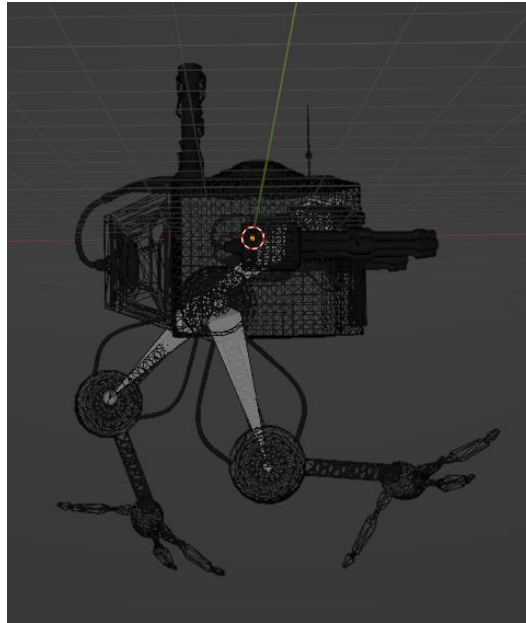
Αυτό μπορεί να γραφεί γενικά ως:

$$J_i = \prod_{j=0}^i (T_j * R_j) * o = A_i * o$$

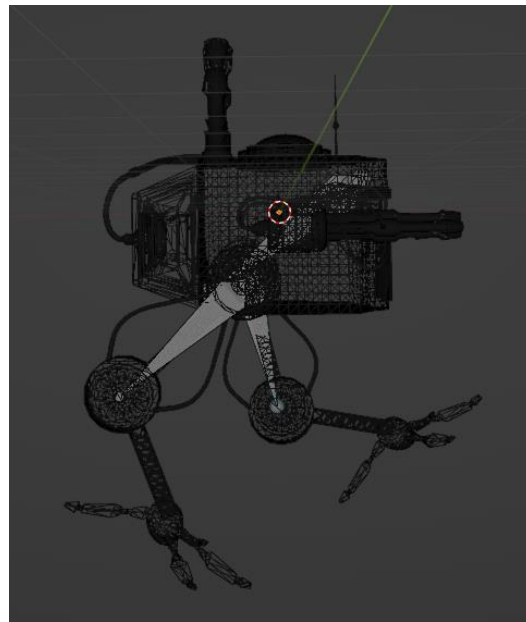


Στο παραπάνω παράδειγμα είναι $i=1$, δηλαδή χρειάζεται να γίνει ένας μετασχηματισμός μέχρι να φτάσουμε στο τοπικό σύστημα συντεταγμένων της ρίζας, ενώ στη συνέχεια υπάρχει ο μετασχηματισμός o , που αντιστοιχεί στον μετασχηματισμό που απαιτείται για να πάμε στο σύστημα συντεταγμένων κόσμου.

Για την κίνηση τώρα, επιλέγονται κάποια βασικά καρέ (poses) και στην συνέχεια μέσω παρεμβολής προκύπτουν οι ενδιάμεσες καταστάσεις.



Σχ.17 pose 1



Σχ.18 pose 2

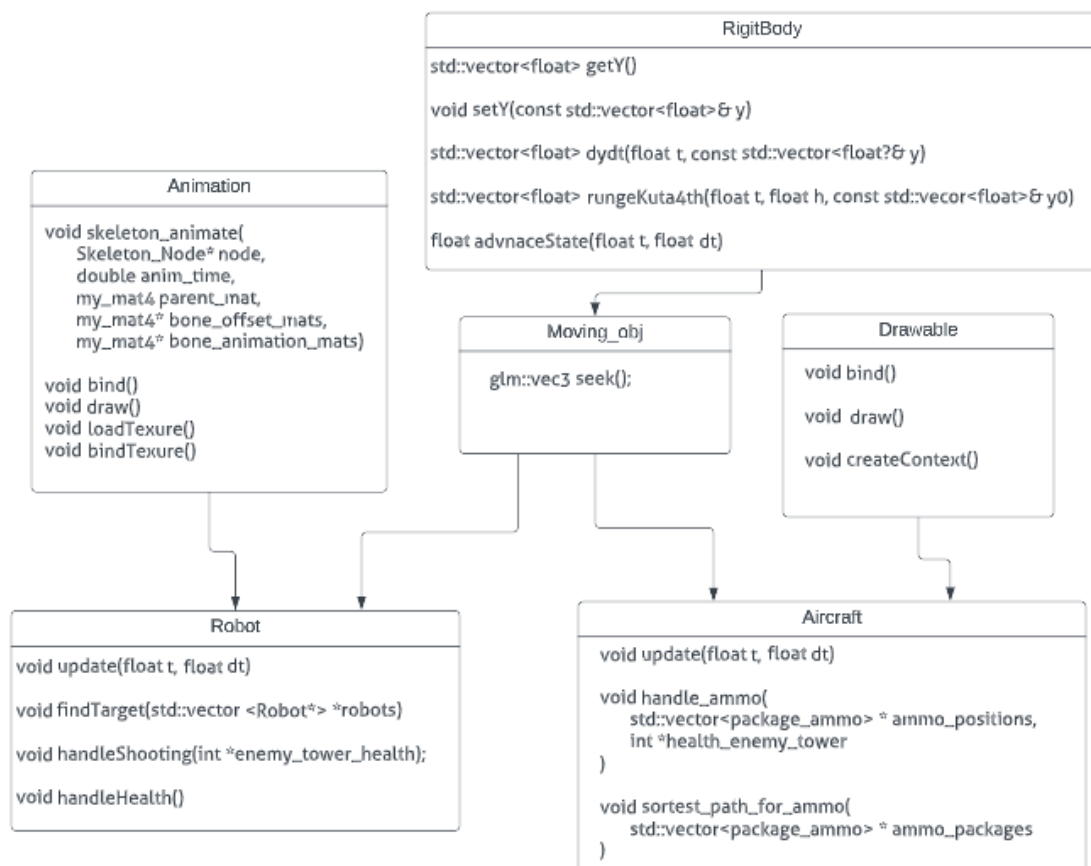
Ύστερα από όλα τα παραπάνω μέσω του blender εξάγεται ένα αρχείο τύπου dae το οποίο περιέχει τις εξής πληροφορίες:

- Vertices
- UVs
- Normal Vectors
- Faces
- Bones
- Weight per vertex per bone
- Parenting of bones
- Poses
- Animation time

Η κλάση `animation` λοιπόν δημιουργήθηκε ώστε να μπορεί να αντλήσει όλες αυτές τις πληροφορίες και να εκτελέσει τους κατάλληλους μετασχηματισμούς ώστε όταν κινείται το `robot` να γίνεται ταυτόχρονα και η κίνηση των ποδιών του.

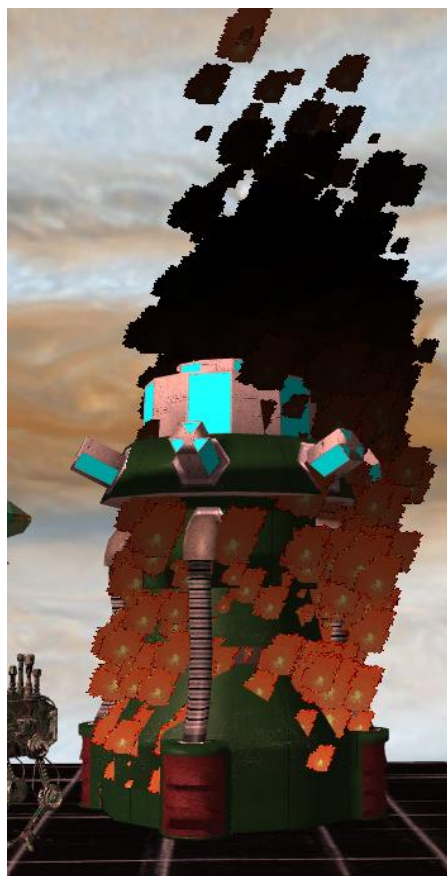
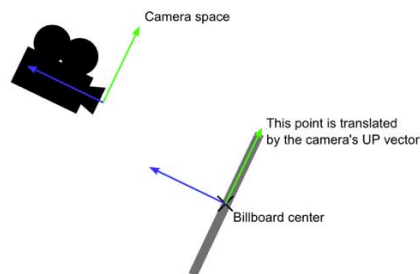
Η κλάση `robot`, είναι η μόνη κλάση που κληρονομεί την κλάση `animation`. Η `robot` κληρονομεί επίσης την `Moving_obj`. Όπως, έχει προαναφερθεί τα `robot`, μπορούν να αμύνονται, εξουδετερώνοντας τα `robot` του αντιπάλου, αλλά και να επιτίθενται στον αντίπαλο πύργο. Η επίθεση γίνεται μόνο όταν ο αντίπαλος δεν έχει διαθέσιμα `robot`. Η κλάση αυτή διαχειρίζεται το `target` του `robot`, και την αναγέννηση του σε περίπτωση που εξουδετερωθεί. Τέλος, διαχειρίζεται το `animation` του `robot` ανάλογα με την ταχύτητα που έχει ως `moving_obj`, δηλαδή όταν το `robot` κινείται με μέγιστη ταχύτητα τότε και ο χρόνος που γίνονται οι αλλαγές στο `animation` (κίνηση ποδιών) πραγματοποιείται με πιο γρήγορο ρυθμό.

Παρακάτω φαίνεται ένα διάγραμμα που δείχνει τον τρόπο με τον οποίο συσχετίζονται οι κλάσεις, που μόλις επεξηγήθηκαν.



Οι κλάσεις `IntParticleEmitter`, `FireEmitter` και `BulletEmitter` αφορούν την τεχνική των `Particles`. Η κλάση `IntParticleEmitter`, υλοποιεί τις διασυνδέσεις που απαιτούνται με την `gpu`, μέσω αυτής δηλαδή στέλνονται τα δεδομένα. Η `FireEmitter` λοιπόν, κληρονομεί την `IntParticleEmitter`, και στόχος της είναι με την λήξη του παιχνιδιού να δημιουργήσει μια

φωτιά γύρω από τον πύργο του παίκτη που έχασε. Πιο αναλυτικά χρησιμοποιείται ένα obj που ουσιαστικά είναι δύο τρίγωνα δημιουργούν ένα quad πάνω στο οποίο έχει τοποθετηθεί ένα texture φωτιάς, το οποίο είναι το particle. Ακόμη, ορίζεται ένα σημείο θέσης του emitter, δηλαδή ένα σημείο από το οποίο θα γεννιούνται όλα τα particles. Κάθε quad που γεννιέται έχει ταχύτητα και επιτάχυνση. Οι παράμετροι αυτοί σε ένα βαθμό είναι προκαθορισμένοι, ωστόσο υπάρχει ένας (τυχαίος) παράγοντας που μπορεί να μεταβάλλει τις τιμές αυτές (σε μικρό βαθμό), ώστε τα particles να μην ακολουθούν την ίδια πορεία, και συνολικά να σχηματίζουν μια φωτιά. Κάθε particle, έχει μια ακόμη παράμετρο που δηλώνει τον χρόνο ζωής του. Ο χρόνος αυτός εξαρτάται από το ύψος στο οποίο βρίσκεται σε σχέση με την αρχική θέση του emitter. Όταν ένα particle πεθαίνει τότε αναγεννιέται ούτως ώστε να υπάρχει σταθερό πλήθος particle (το πλήθος των particles στον fire emitter είναι 2000). Επίσης, αξίζει να σημειωθεί ότι χρησιμοποιείται billboard τεχνική. Η ιδέα του billboard είναι ότι με βάση την κίνηση της κάμερας το quad μετακινείται ώστε το normal vector του quad να είναι παράλληλο με το διάνυσμα κατεύθυνσης που κοιτάει η camera.

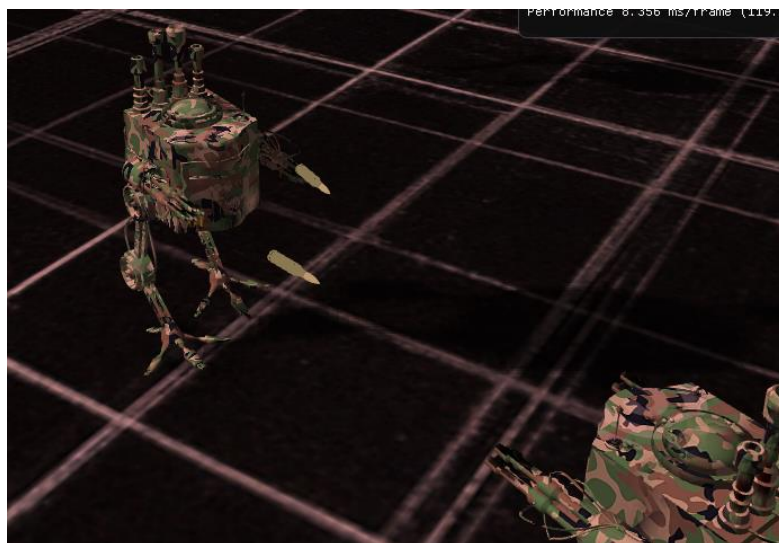


Σχ.19 πύργος με φωτιά από 1^η οπτική

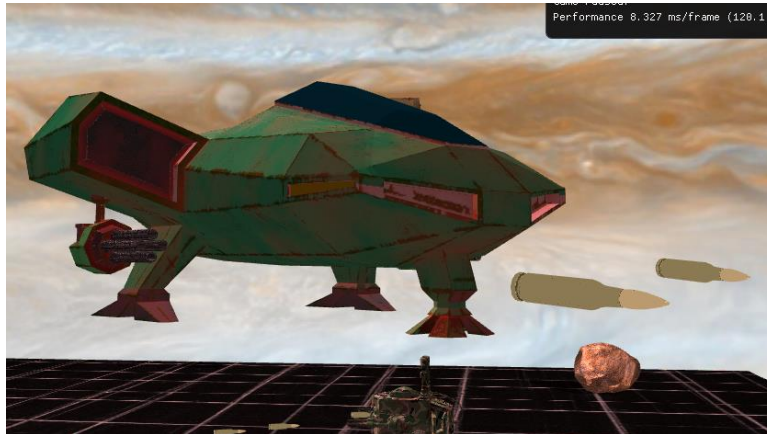


Σχ.20 πύργος με φωτιά από 2^η οπτική

Η BulletEmitter είναι μια κλάση που κατασκευάστηκε για να αναπαραστήσει τις σφαίρες των aircrafts και των robots. Και εκείνη κληρονομεί την IntParticleEmitter. Η θέση του emitter, δηλαδή το σημείο από το οποίο γεννιέται ένα particle εξαρτάται από την θέση του aircraft/robot. Ακόμη, η ταχύτητα και η επιτάχυνση που αποκτούν τα particles δεν είναι τυχαία αλλά καθορισμένη. Το διάνυσμα της ταχύτητας των particles εξαρτάται από τον στόχο του aircraft/robot.



Σχ.21 robot shooting



Σχ.22 aircraft shooting

Τέλος η κλάση **Light** η οποία χρησιμοποιείται για τον φωτισμό. Με την κλάση Light δημιουργούμε μια φωτεινή πηγή τύπου pointing light η οποία βρίσκεται στον χώρο υπό μορφή σφαίρας. Ο χρήστης έχει την δυνατότητα να μετακινήσει την σφαίρα πατώντας το πλήκτρο 1 και με τα πλήκτρα:

- I -> αυξάνει το y.
- K -> μειώνει το y.
- L -> αυξάνει το x.
- J -> μειώνει το x.
- U -> μειώνει το z.
- O -> αυξάνει το z.

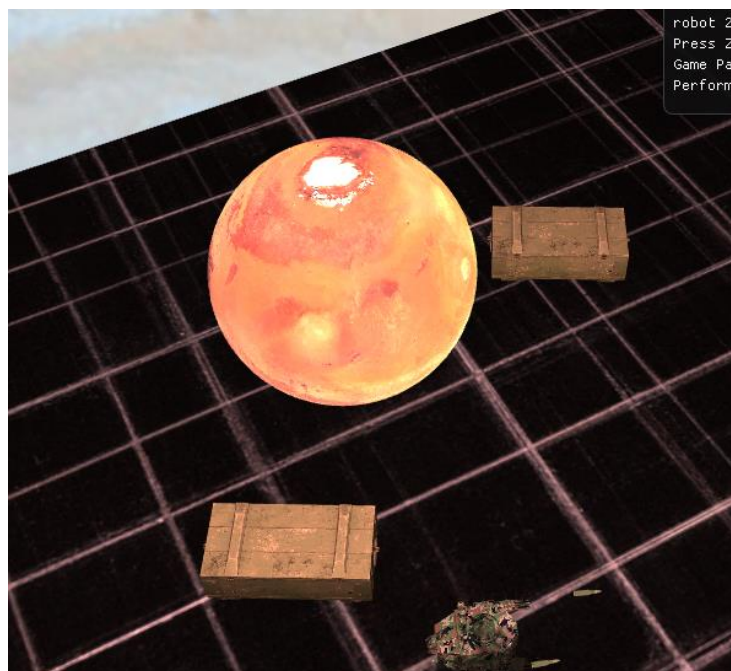
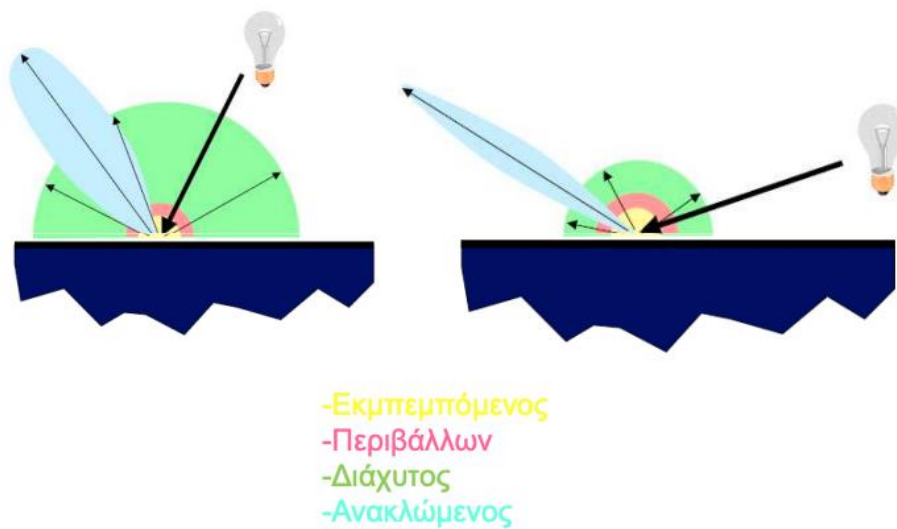
Σχ. Φωτεινή πηγή

Για τον φωτισμό χρησιμοποιήθηκε το μοντέλο του Phong. Το μοντέλο αυτό χρησιμοποιεί μαθηματικούς όρους που δεν πηγάζουν από φυσικούς νόμους, ωστόσο δίνει ιδιαίτερα ικανοποιητικά αποτελέσματα. Η κατανομή αμφίδρομης ανακλαστικότητας (ΚΑΑ) του μοντέλου αυτού συσχετίζει την προσπίπτουσα φωτεινή ένταση από την κατεύθυνση της φωτεινής πηγής με την ανακλώμενη φωτεινή ένταση, για ένα σημείο αντικειμένου p. Ακόμη, υπολογίζει την ορατή ένταση ως άθροισμα 4 όρων.

- Φωτισμός εκπομπής (Ie)
- Περιβάλλον φωτισμός (Ig)
- Διάχυτη ανάκλαση (Id)
- Κατοπτρική ανάκλαση (Is)

Ο όρος εκπομπής Ie αφορά αυτόφωτα αντικείμενα. Ο όρος περιβάλλοντος φωτισμού Ig αντισταθμίζει το γεγονός ότι το μοντέλο Phong είναι ένα μοντέλο τοπικού φωτισμού (χωρίς αυτό τον όρο μια επιφάνεια που δεν φωτίζεται απευθείας από μια πηγή θα εμφανίζονταν τελείως σκοτεινή). Οι όροι διάχυτης και κατοπτρικής ανάκλασης έχουν σχέση με την γωνία που πέφτει το φως πάνω στο σημείο φωτισμού, ακόμη υπάρχουν συντελεστές kd,ks που επηρεάζουν το τους όρους Id,Is αντιστοίχως ανάλογα με τα χαρακτηριστικά της επιφάνειας του αντικειμένου. Τέλος, ο όρος της κατοπτρικής ανάκλασης εξαρτάται από την διεύθυνση

παρατήρησης ενώ ο όρος της διάχυτης ανάκλασης όχι. Το παραπάνω μοντέλο υλοποιείται στους shaders.



Σχ.23 light

Μια επιπλέον τεχνική που υλοποιήθηκε στο project είναι το **shadow mapping** για τις σκιές. Πιο αναλυτικά το shadow mapping υλοποιείται μέσω ενός texture. Στο texture αυτό κρατείται το depth value για κάθε fragment από το διάνυσμα που κοιτάει το direction light. Με αυτή την πληροφορία μπορεί να γίνει αντιληπτό ποια fragments έρχονται σε πρώτη επαφή με το φως και ποια όχι. Τα fragments που έχουν άμεση επαφή με το φως θα φωτίζονται πλήρως ενώ τα fragments που βρίσκονται από πίσω θα σκιάζονται, δηλαδή θα μειώνεται το ποσοστό φωτεινότητας τους.



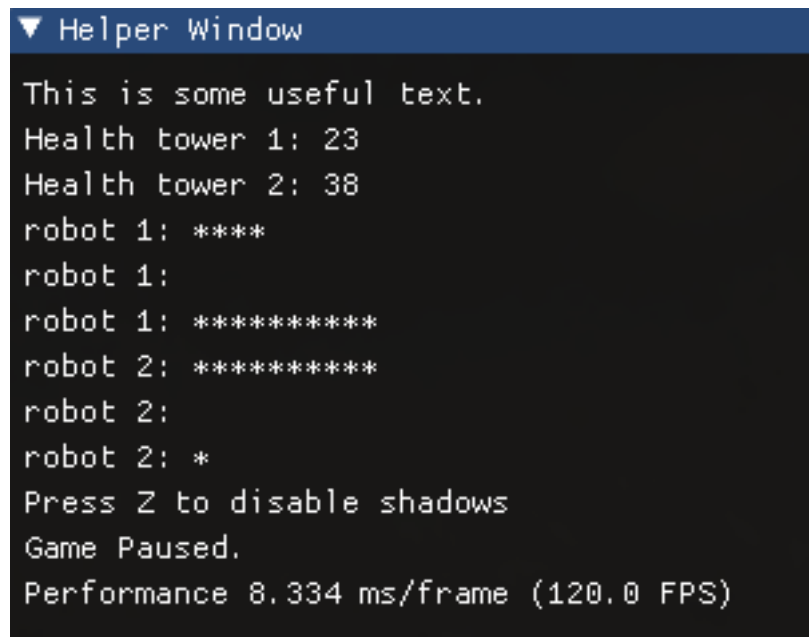
Σχ.24 with shadows



Σχ.25 without shadows

Τέλος, έχει υλοποιηθεί ένα gui από το οποίο ο χρήστης μπορεί να πάρει πληροφορίες σχετικά με το παιχνίδι. Πιο συγκεκριμένα μπορεί να δει τις ζωές των δύο πύργων, και τις ζωές των robot όταν αυτά ενεργοποιηθούν από τον παίκτη. Ακόμη ο χρήστης έχει την δυνατότητα να πατήσει παύση του παιχνιδιού με το πλήκτρο P. Μπορεί επίσης με το πλήκτρο Z να

απενεργοποιήσει τις σκιές (ή να τις ενεργοποιήσει σε περίπτωση που τις έχει απενεργοποιήσει). Τέλος, με την λήξη του παιχνιδιού στο gui αναγράφεται ποιος είναι ο νικητής του παιχνιδιού.

A screenshot of a 'Helper Window' with a dark blue title bar. The window contains white text on a black background. The text displays game statistics: 'This is some useful text.', 'Health tower 1: 23', 'Health tower 2: 38', 'robot 1: ****', 'robot 1:', 'robot 1: *****', 'robot 2: *****', 'robot 2:', 'robot 2: *', 'Press Z to disable shadows', 'Game Paused.', and 'Performance 8.334 ms/frame (120.0 FPS)'.

```
▼ Helper Window  
  
This is some useful text.  
Health tower 1: 23  
Health tower 2: 38  
robot 1: ****  
robot 1:  
robot 1: *****  
robot 2: *****  
robot 2:  
robot 2: *  
Press Z to disable shadows  
Game Paused.  
Performance 8.334 ms/frame (120.0 FPS)
```

Σχ.26 gui

Κώδικας: <https://github.com/eefthymiou/TowerDefense.git>