ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ
ΤΕΙ ΗΠΕΙΡΟΥ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε

Μαθαίνοντας προγραμματισμό σε περιβάλλον ανοικτών μαθημάτων (MOOC) με χρήση παιχνιδιών

ΒΟΥΖΑ ΕΛΕΝΗ ΑΘΑΝΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ ΚΑΘΗΓΗΤΗΣ:ΛΙΑΡΟΚΑΠΗΣ ΔΗΜΗΤΡΙΟΣ

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΙΣΑΓΩΓΗ

Σκοπός της συγκεκριμένης πτυχιακής εργασίας είναι η εξοικείωση του αναγνώστη με τα ελεύθερα διαδικτυακά μαθήματα ή αλλιώς MOOC. Συγκεκριμένα μέσα από την διαδικτυακή πλατφόρμα COURSERA η οποία φιλοξενεί πάνω από 1500 μαθήματα παρουσιάζεται το μάθημα «Εισαγωγή στον διαδραστικό προγραμματισμό με PYTHON».

Στόχος του μαθήματος είναι να διδάξει στους φοιτητές την γλώσσα προγραμματισμού PYTHON έτσι ώστε να υλοποιήσουν τα δικά τους προγράμματα μέσω ενός προγράμματος διαδραστικού προγραμματισμού το CODESKULPTOR.

Πιο αναλυτικά το μάθημα «Εισαγωγή στον διαδραστικό προγραμματισμό με PYTHON» διάρκειας εννέα εβδομάδων διδάσκεται μέσω βίντεο-διαλέξεων τα οποία έχει στην διάθεση του ο φοιτητής οποιαδήποτε στιγμή. Με το πέρας των διαλέξεων ο φοιτητής πρέπει να συμπληρώσει δύο κουίζ ερωτήσεων και ασκήσεων και να υλοποιήσει μία προγραμματιστική άσκηση.

Η πτυχιακή εργασία αποτελείται από εννέα κεφάλαια. Κάθε κεφάλαιο αντιστοιχεί σε μία εβδομάδα του μαθήματος Στην αρχή κάθε κεφαλαίου παρουσιάζονται οι εντολές της PYTHON με τις οποίες είναι συμβατές το CODESKULPTOR και παρατίθενται παραδείγματα για τον τρόπο χρησιμοποίησης αυτών των εντολών. Στο τέλος κάθε κεφαλαίου δίνεται μια πρακτική άσκηση για την δημιουργία παιχνιδιών με την χρησιμοποίηση των εντολών του CODESKULPTOR.

INTRODUCTION

This particular thesis intends to familiarize the reader with the massive open online courses or else MOOC. Through the online COURSERA platform which hosts over than 1500 online courses the course Introduction to Python is represented.

This course aims to teach the students the programming language PYTHON so that they implement their own programs through CODESKULPTOR program.

More thoroughly the course which lasts nine weeks is taught through videos which are at the student’s disposal anytime. At the end of the videos the student must complete two quizzes with questions and exercises and implement one programming exercise.

This thesis consists of nine chapters. Each chapter corresponds to one week of the course. At the start of the each chapter the commands of PYTHON which are compatible with CODESKULPTOR are being introduced and examples of the way of using these commands are stated. At the end of each chapter one practical exercise is given so that a game can be created with the commands of CODESKULPTOR.

1. MASSIVE OPEN ONLINE COURSES (MOOC)

Τα " Μαζικά Ελεύθερα Διαδικτυακά Μαθήματα" ή MOOC είναι διαδικτυακά μαθήματα που έχουν ως σκοπό την μαζική συμμετοχή και ανοικτή πρόσβαση στη γνώση μέσω του διαδικτύου. Ο καθένας με έναν υπολογιστή και σύνδεση στο διαδίκτυο μπορεί να εγγραφεί και να παρακολουθήσει διαδικτυακά οποιοδήποτε μάθημα. Τα MOOC παρέχονται δωρεάν και είναι προσβάσιμα για όλους ανεξαρτήτως εκπαιδευτικού υπόβαθρου.

Η διδασκαλία σε MOOC μαθήματα γίνεται μέσω εβδομαδιαίων βίντεο-διαλέξεων, φορμών συμπλήρωσης ερωτήσεων -ασκήσεων(κουίζ) και εργασιών. Η διδασκαλία των MOOC μαθημάτων δεν διαφέρει καθόλου σε σχέση με παραδοσιακά πανεπιστημιακά μαθήματα διότι έχουν συγκεκριμένες ημερομηνίες έναρξης και λήξης μαθημάτων και τηρούνται αυστηρά οι διορίες παράδοσης των εργασιών. Τα MOOC δίνουν την δυνατότητα αλληλεπίδρασης φοιτητών με καθηγητές όπως επίσης και με τους συμφοιτητές τους μέσα από διαδικτυακές ομάδες συζητήσεων(forum) με εξαίρεση ίσως κάποιων MOOC μαθημάτων τα οποία συγκεντρώνουν μεγάλο αριθμό συμμετεχόντων όπου καθίσταται αδύνατη η επικοινωνία μεταξύ καθηγητών φοιτητών.

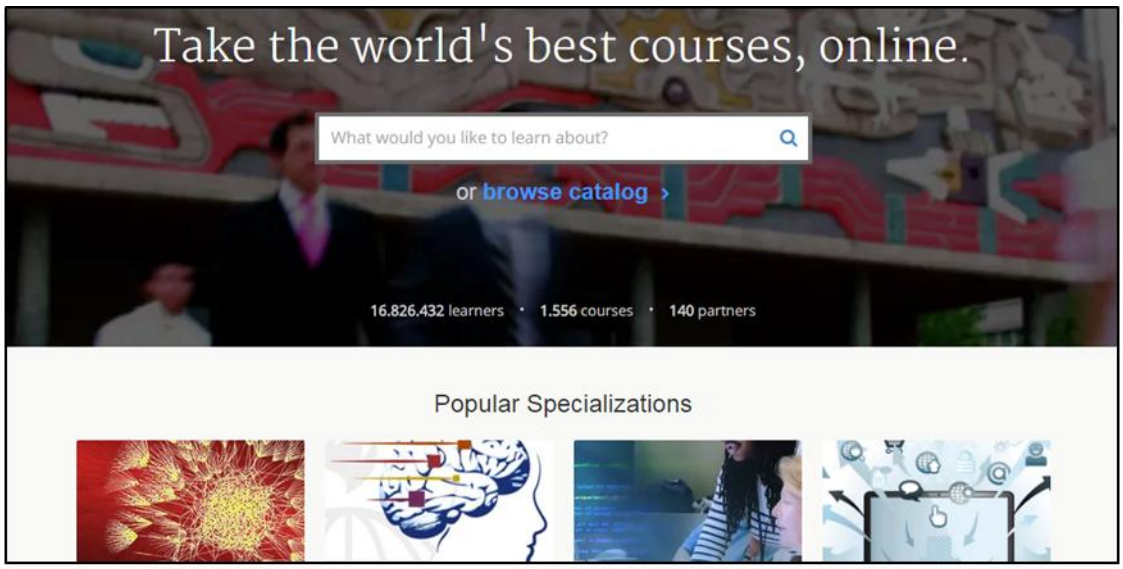
Η αξιολόγηση των εργασιών γίνεται με δύο τρόπους, είτε με αυτόματη αξιολόγηση των εργασιών και των φορμών συμπλήρωσης ερωτήσεων από την πλατφόρμα είτε με αξιολόγηση των εργασιών των φοιτητών από τους άλλους φοιτητές. Στη δεύτερη περίπτωση δίνονται πάντα οδηγίες από τους υπεύθυνους καθηγητές των μαθημάτων για τον σωστό τρόπο αξιολόγησης των εργασιών. Συνήθως η αξιολόγηση κάθε εργασίας γίνεται από περισσότερους από ένα φοιτητές για να γίνει εφικτή η αξιοκρατικότερη βαθμολόγηση κάθε εργασίας.

Με την επιτυχή ολοκλήρωση των μαθημάτων τα περισσότερα πανεπιστήμια πλην κάποιων εξαιρέσεων παρέχουν πιστοποιητικά παρακολούθησης, ορισμένα μάλιστα παρέχουν επίσημα πιστοποιητικά με το ορόσημο του εκάστοτε πανεπιστημίου έναντι κάποιου χρηματικού αντίτιμου.

1.1 ΔΙΕΘΝΗΣ ΣΥΝΕΡΓΑΣΙΕΣ ΜΑΖΙΚΩΝ ΕΛΕΥΘΕΡΩΝ ΜΑΘΗΜΑΤΩΝ

Πολλές διαδικτυακές πλατφόρμες προσφέρουν μαθήματα MOOC,οι τρείς μεγαλύτεροι πάροχοι MOOC μαθημάτων είναι το COURSERA,EDX και UDACITY.

Πλατφόρμα Coursera(http://www.coursera/org)

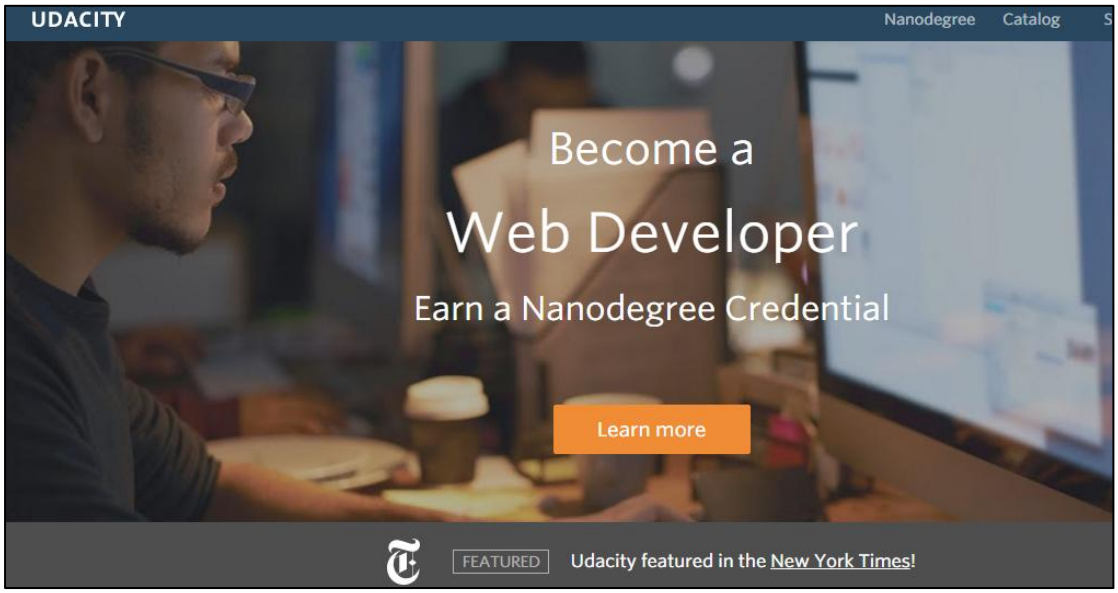


Το COURSERA είναι μία κερδοσκοπικού τύπου κοινοπραξία κορυφαίων πανεπιστημιακών ιδρυμάτων όπως το BROWN UNIVERSITY, UNIVERSITY OF MICHIGAN και άλλα. Προσφέρει πάνω από 1500 μαθήματα σε εννέα κλάδους όπως πληροφορική, μαθηματικά και μηχανολογία .Τα μαθήματα είναι προσβάσιμα στους φοιτητές με την προϋπόθεση εγγραφής.Για την συμμετοχή στα μαθήματα δεν απαιτείται η καταβολή διδάκτρων ενώ στην περίπτωση επιτυχούς παρακολούθησης γίνεται εφικτή η απόκτηση πιστοποιητικού το οποίο δεν αναγνωρίζεται επίσημα από το πανεπιστήμιο αλλά υπογράφεται μόνο από τους καθηγητές του μαθήματος. Ωστόσο ορισμένα πανεπιστήμια προσφέρουν επίσημα πιστοποιητικά με την καταβολή κάποιου χρηματικού ποσού. Οι φοιτητές έχουν πρόσβαση σε ψηφιακό υλικό όπως επίσης σε διαφάνειες και συγγράμματα τα οποία διατίθενται από τους καθηγητές Οι φοιτητές καθορίζουν τον τόπο και τον χρόνο υλοποίησης των εργασιών μέσα στα χρονικά πλαίσια του μαθήματος. Η διδασκαλία γίνεται μέσω βίντεο-διαλέξεων, Για την επιτυχή ολοκλήρωση των μαθημάτων απαιτείται η διεκπεραίωση φορμών συμπλήρωσης ερωτήσεων και εργασιών .Το σύνολο των μαθημάτων διδάσκεται στην αγγλική γλώσσα πλην ορισμένων τα οποία προσφέρουν μαθήματα στην μητρική τους γλώσσα. Ένα πλεονέκτημα της πλατφόρμας COURSERA είναι ότι δίνεται η δυνατότητα στους φοιτητές να συνεργάζονται με τους καθηγητές το τεχνικό προσωπικό όπως και τους φοιτητές για την επίλυση των εργασιών μέσα στα πλαίσια του κώδικα δεοντολογίας.

Πλατφόρμα Edx(https://www.edx.org)

Το EDX είναι μία κοινοπραξία μη κερδοσκοπικού τύπου των πανεπιστημίων HARVARD και MIT. Το μαθήματα είναι προσβάσιμα στους φοιτητές με την προϋπόθεση εγγραφής. Για την συμμετοχή στα μαθήματα δεν απαιτείται η καταβολή διδάκτρων ενώ στην περίπτωση επιτυχούς παρακολούθησης γίνεται εφικτή η απόκτηση πιστοποιητικού. Δίνεται η δυνατότητα αλληλεπίδρασης μεταξύ των φοιτητών όπως και με τους καθηγητές σε κατάλληλο χώρο που ορίζεται από τους καθηγητές. Οι φοιτητές έχουν πρόσβαση σε βίντεο-διαλέξεις όπως και σε διαφάνειες και συγγράμματα.Οι καθηγητές καθορίζουν τα χρονικά πλαίσια υλοποίησης των εργασιών. Το EDX προσφέρει πάνω από 500 διαδικτυακά μαθήματα με ένα μεγάλο σύνολο αυτών να διδάσκονται έκτος από τα αγγλικά σε άλλες οκτώ γλώσσες όπως κινέζικα, ισπανικά και ιταλικά.

Πλατφόρμα Udacity(https://www.udacity.com/)



Το UDACITY είναι μία κοινοπραξία καθηγητών του STANFORD UNIVERSITY κερδοσκοπικού τύπου χωρίς την σύμπραξη πανεπιστημίων αλλά ιδιωτικών εταιριών. Απευθύνεται ως επί το πλείστον σε επαγγελματίες και προσφέρει μαθήματα στον τομέα της πληροφορικής. Τα μαθήματα είναι προσβάσιμα στους φοιτητές με την προϋπόθεση εγγραφής και την καταβολή διδάκτρων. Δίνεται ωστόσο η δυνατότητα στους φοιτητές να συμμετάσχουν σε μία δοκιμαστική περίοδο εκμάθησης προτού καταβάλουν οποιοδήποτε χρηματικό ποσό. Για την επιτυχή ολοκλήρωση των μαθημάτων απαιτείται η διεκπεραίωση εργασιών και φορμών συμπλήρωσης ερωτήσεων. Το πλεονέκτημα του UDACITY είναι ότι δεν τίθεται αυστηρό χρονοδιάγραμμα διεκπεραίωσης των εργασιών.

1.2 ΠΛΕΟΝΕΚΤΗΜΑΤΑ ΤΩΝ MOOC

1.Είναι προσβάσιμα σε όποιον έχει την δυνατότητα σύνδεσης στο Internet.

2.Δεν χρειάζονται εξειδικευμένες γνώσεις υπολογιστή.

3.Διδάσκονται κατά το πλείστον στα αγγλικά αλλά και σε άλλες γλώσσες.

4.Βελτιώνουν το εκπαιδευτικό υπόβαθρο.

5.Είναι προσβάσιμα για όλους ανεξαρτήτου ηλικίας.

6.Διαρκούν ένα μικρό χρονικό διάστημα.

7.Αναπτύσεται η συνεργασία και η ομαδικότητα των φοιτητών .

1.3 ΜΕΛΛΟΝΤΙΚΕΣ ΠΡΟΣΔΟΚΙΕΣ ΤΩΝ MOOC

Σκοπός της δημιουργίας των διαδικτυακών μαθημάτων ήταν η παροχή γνώσεων σε όλους ανεξαρτήτου ηλικίας και κοινωνικού υπόβαθρου.Καποιοι μάλιστα πιστεύουν ότι τα MOOC θα καταργήσουν μελλοντικά τα πανεπιστημιακά ιδρύματα.Τις χρονιές 2013-2014 η δημοτικότητα των MOOC έφτασε στο ζενίθ της όταν εκατομμύρια άνθρωποι,οι οποίοι δεν είχαν την δυνατότητα να παρακολουθήσουν μαθήματα σε κάποια από τα πρωτοκλασάτα πανεπιστήμια του κόσμου, στράφηκαν στα MOOC μαθήματα με την ελπίδα να αποκτήσουν καλύτερες γνώσεις ή καλύτερη δουλεία. Συγκεκριμένα μελέτες έδειξαν ότι οι περικοπές που έγιναν σε εκπαιδευτικά ιδρύματα στην Ελλάδα και την Ισπανία λόγω της οικονομικής κρίσης οδήγησαν τους φοιτητές που παρακολουθούσαν μαθήματα MOOC να πληρώνουν δίδακτρα για την απόκτηση πιστοποιητικού ελπίζοντας σε καλύτερη επαγγελματική αποκατάσταση.Παρόλου που ο αριθμός των εγγεγραμμένων φοιτητών έχει αυξηθεί ραγδαία τα τελευταία χρόνια το μέλλον το MOOC είναι αβέβαιο την στιγμή μάλιστα που χιλιάδες φοιτητές

ξεγράφονται από τα μαθήματα ή δεν τα ολοκληρώνουν ποτέ.Ένας σημαντικός λόγος για αυτό το φαινόμενο είναι η έλλειψη κινήτρου.Δυστυχώς τα MOOC προσφέρουν αναγνωρισμένα πιστοποιητικά μόνο με την καταβολή χρηματικού ποσού,ποσό που οι περισσότεροι φοιτητές δεν είναι διατεθειμένοι δώσουν.Ισως μελλοντικά ξεπεραστούν τέτοιου είδους προβλήματα με περισσότερη χρηματοδότηση από τα πανεπιστήμια έτσι ώστε να δημιουργηθούν κέντρα εξέτασης σε όλο τον κόσμο.Παρα τις αμφιβολίες που υπάρχουν για την αποδοτικότητα των MOOC το μόνο σίγουρο είναι ότι όλη αυτή η γνώση είναι διαθέσιμη για όλους.

1.4 Εισαγωγή στον διαδραστικό προγραμματισμό με PYTHON (An Introduction to Interactive Programming in Python)

Το μάθημα «Εισαγωγή στον διαδραστικό προγραμματισμό με PYTHON» διδάσκεται στην πλατφόρμα COURSERA από το RICE UNIVERSITY.Σκοπός του μαθήματος είναι να βοηθήσει φοιτητές με λίγη ή καθόλου γνώση προγραμματισμού να σχεδιάσουν εφαρμογές διαδραστικού χαρακτήρα στην γλώσσα προγραμματισμού PYTHON.

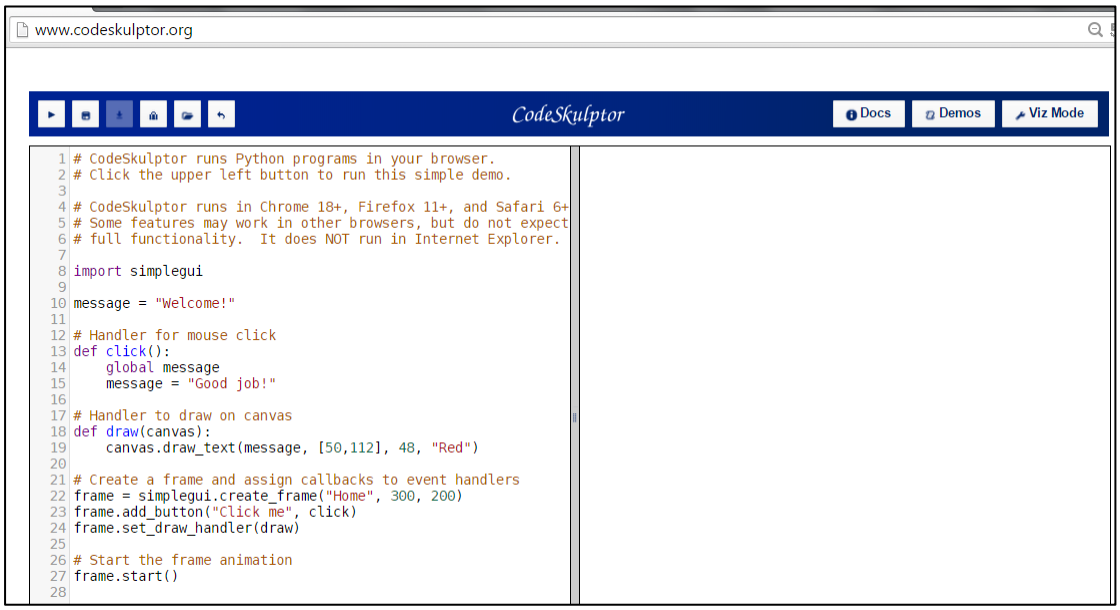
Η διάρκεια του μαθήματος είναι οχτώ εβδομάδες. Κάθε βδομάδα ο φοιτητής είναι υποχρεωμένος να παρακολουθήσει ολιγόλεπτες βίντεο-διαλέξεις.Για την επιτυχή παρακολούθηση του μαθήματος ο φοιτητής πρέπει να υλοποιήσει δύο εβδομαδιαία κουίζ και μία προγραμματιστική άσκηση.Στο τέλος του μαθήματος ο φοιτητής έχει την δυνατότητα να αποκτήσει πιστοποίηση παρακολούθησης αφού έχει ολοκληρώσει με επιτυχία το 70% των ασκήσεων.

Δίνεται η δυνατότητα συνεργασίας των φοιτητών μεταξύ τους πάντα στα πλαίσια δεοντολογίας μέσω ενός forum επικοινωνίας που βρίσκεται στην πλατφόρμα του μαθήματος και διαφόρων ομάδων μελέτης. Για πιο εύκολη κατανόηση της PYTHON αναπτύχθηκε από τους καθηγητές του μαθήματος ένα περιβάλλον προγραμματισμού, το CODESKULPTOR το οποίο είναι προσβάσιμο μέσω διαδικτύου.

1.5 ΓΝΩΡΙΜΙΑ ΜΕ ΤΟ CODESKULPTOR

Το CODESKULPTOR είναι ένα διαδραστικό περιβάλλον προγραμματισμού σε Python.Το CODESKULPTOR επιτρέπει στον κώδικα να τρέξει απευθείας από τον browser. Το περιβάλλον του CODESKULPTOR αναπτύχθηκε από τον Scott Rixner καθηγητή του πανεπιστήμιου Rice και είναι διαθέσιμο σε όλους τους χρήστες που έχουν πρόσβαση στο διαδίκτυο μέσω της ιστοσελίδας «www.codeskulptor.org».

Codeskulptor (www.codeskulptor.org)



Το περιβάλλον του CODESKULPTOR τρέχει σε browser και είναι συμβατό με το Chrome18+ ,Firefox11+ και Safari.Σε άλλους browsers ενδέχεται κάποια χαρακτηριστικά του CODESKULPTOR να μην είναι διαθέσιμα ενώ δεν είναι συμβατό με τον Internet Explorer.Κάποιες από τις πιο σημαντικές λειτουργίες του CODESKULPTOR περιλαμβάνουν:

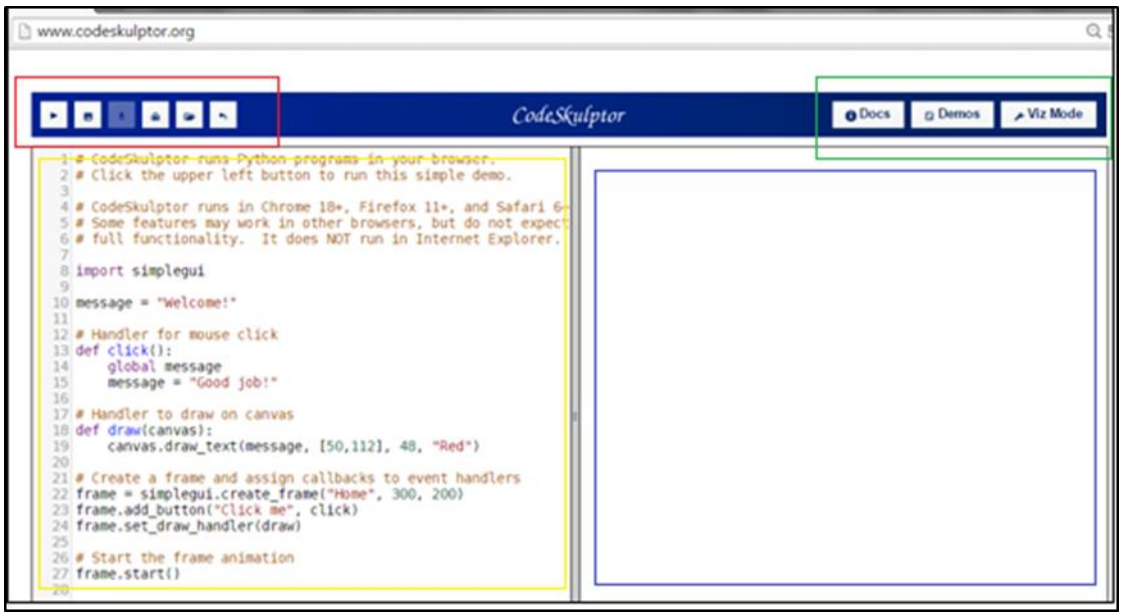
α. Σύνταξη προγραμμάτων διαδραστικού χαρακτήρα.

β. Επεξεργασία και αποθήκευση προγραμμάτων.

γ. Εμφάνιση μηνυμάτων «λάθους» για τον εντοπισμό σφαλμάτων.

δ.Visualization mode.(Οπτική απεικόνιση της λειτουργίας του προγράμματος).

1.5.1 ΛΕΙΤΟΥΡΓΙΕΣ ΤΟΥ CODESKULPTOR



Control area: Η περιοχή στο κόκκινο πλαίσιο ονομάζεται περιοχή ελέγχου. Η περιοχή ελέγχου αποτελείται από buttons με τα οποία ο χρήστης ελέγχει την λειτουργία του προγράμματος.

Editor: Η περιοχή στο κίτρινο πλαίσιο αποτελεί τον editor του Codeskulptor όπου συντάσσεται το πρόγραμμα.

Documentation area: Η περιοχή στο πράσινο πλαίσιο αποτελείται από την καρτέλα «Docs» η οποία περιέχει όλες τις βιβλιοθήκες και τις εντολές του Codeskulptor, την καρτέλα Demos στην οποία παρουσιάζονται έτοιμα προγράμματα σε Python και την καρτέλα Viz Mode με την οποία απεικονίζεται βήμα-βήμα η λειτουργία του προγράμματος.

Exit area: Η περιοχή στο μπλε πλαίσιο ονομάζεται περιοχή εξόδου του προγράμματος όπου απεικονίζεται η έξοδος του προγράμματος σε μορφή κειμένου.

Buttons περιοχής ελέγχου

 Run Button

Το run button εκτελεί το πρόγραμμα και εμφανίζει την έξοδο του προγράμματος στην περιοχή εξόδου.

 Save Button

Το save button δίνει αυτόματα μία διεύθυνση URL στο πρόγραμμα και το αποθηκεύει σε ένα απομακρυσμένο κέντρο δεδομένων. Με αυτόν τον τρόπο δίνεται η δυνατότητα

στον χρήστη να αποκτήσει πρόσβαση στο πρόγραμμα του από οπουδήποτε αρκεί να διαθέτει την διεύθυνση URL και πρόσβαση στο διαδίκτυο.

 Fresh Url Button

Το Fresh url button ανανεώνει την διεύθυνση URL του προγράμματος.

 Open Local Button

Το Open Local Button ανοίγει τοπικά αρχεία του Η/Υ.

 Reset Button

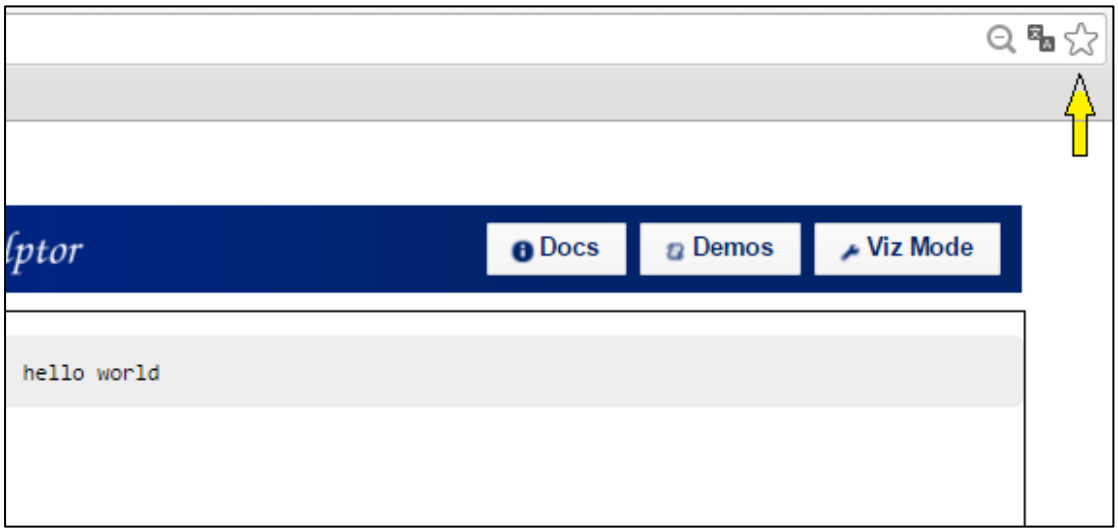
Το Reset button διαγράφει την έξοδο του προγράμματος στην περιοχή exit area.

1.5.2 ΑΠΟΘΗΚΕΥΟΝΤΑΣ ΣΤΟ CODESKULPTOR ΜΕ ΧΡΗΣΗ ΣΕΛΙΔΟΔΕΙΚΤΩΝ

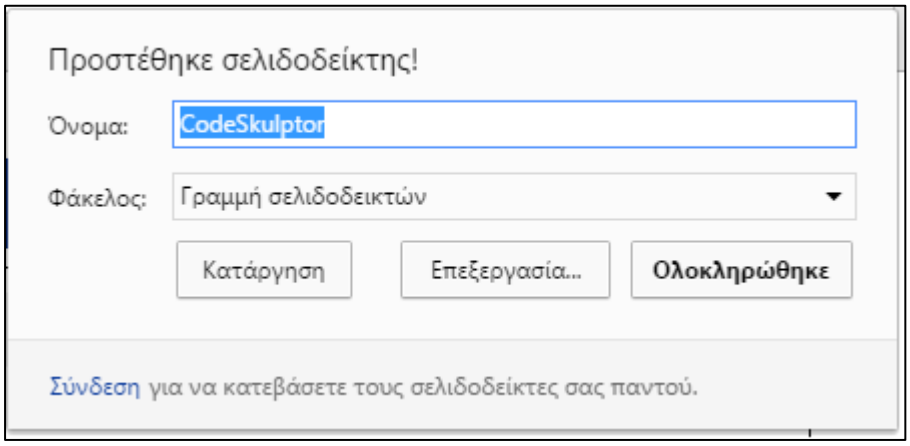
Όπως αναφέρθηκε προηγουμένως τα προγράμματα στο Codeskulptor αποθηκεύονται σε ένα απομακρυσμένο κέντρο δεδομένων και όχι στον υπολογιστή μας. Η διαχείριση των προγραμμάτων καθίσταται πιο εύκολη με την αποθήκευση τους σε σελιδοδείκτες του browser. Ας δούμε με ένα παράδειγμα την χρήση σελιδοδεικτών στο Chrome.

1.Αποθηκεύση του προγράμματος με την χρήση του Save Button.Αυτόματα δημιουργείται μια διεύθυνση url.

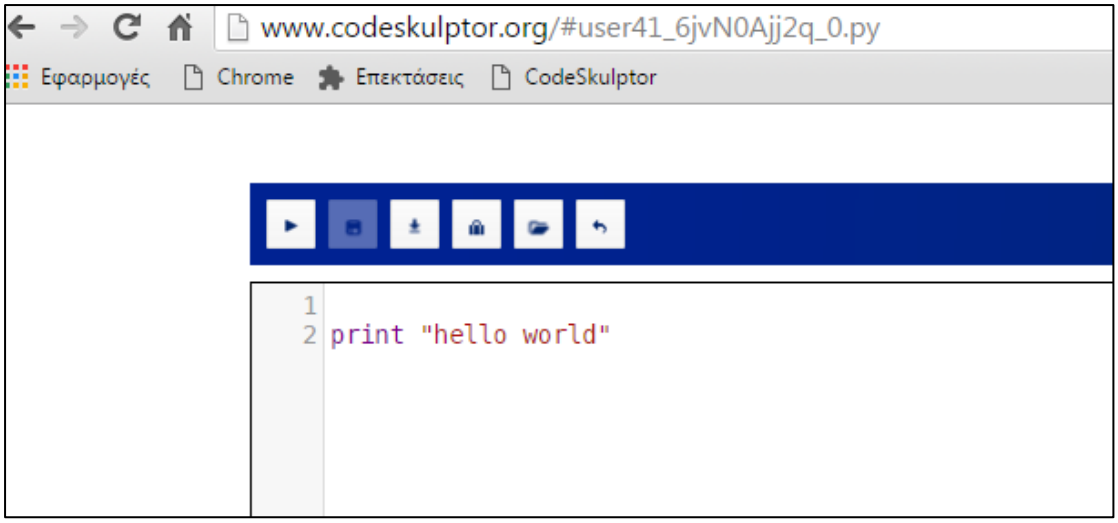
2.Κάνουμε κλικ στην επιλογή «Προσθήκη αυτής της σελίδας στους σελιδοδείκτες» στο επάνω μέρος της μπάρας του Chrome όπως φαίνεται στην παρακάτω εικόνα.



3. Εμφανίζεται μήνυμα προσθήκης του σελιδοδείκτη



4.Εμφάνιση ετικέτας στην μπάρα του Chrome με το όνομα του σελιδοδείκτη



2. ROCK-PAPER-SCISSORS-LIZARD-SPOCK

Την πρώτη εβδομάδα του μαθήματος υλοποιήθηκε το Rock-Paper-Scissors-Lizard-Spock Για την υλοποίηση του συγκεκριμένου παιχνιδιού ο αναγνώστης πρέπει να εξοικειωθεί με την έννοια της συνάρτησης όπως και με τις εντολές του Codeskulptor που είναι αναγκαίες για την υλοποίηση του παιχνιδιού. Συγκεκριμένα σε αυτό το κεφάλαιο αναλύονται η λειτουργία των συναρτήσεων στην Python όπως και τα random module που προσφέρει το Codeskulptor για την δημιουργία συναρτήσεων περιοδικότητας.

2.1 ΣΥΝΑΡΤΗΣΕΙΣ (FUNCTIONS)

Στην PYTHON μια συνάρτηση λειτουργεί όπως μια μαθηματική συνάρτηση με την μόνη διαφορά ότι η προγραμματιστική συνάρτηση επιστρέφει το αποτέλεσμα της με την μορφή αντικειμένου. Ο κώδικας της εκτελείται μόνο όταν καλείται η συνάρτηση και μπορούμε να την καλέσουμε όσες φορές χρειαστεί.

Συντάσσοντας μια συνάρτηση :

```
def a_func (var0, var1, …):  …
```

Στο παρακάτω παράδειγμα η συνάρτηση triangle_area υπολογίζει το εμβαδόν τριγώνου.

```
def triagle_area (base, height):
    area= (1.0 /2) *base *height
    return area
```

Η εντολή **def triangle_area** δημιουργεί την συνάρτηση triangle_area .Η συνάρτηση παίρνει σαν παραμέτρους την βάση και το ύψος του τριγώνου. Αφού υπολογίζεται το εμβαδόν της συνάρτησης επιστρέφεται με την εντολή return.Αν τρέξουμε το παραπάνω κομμάτι κώδικα στο Codeskulptor δεν θα εμφανιστεί τίποτα. Πρώτα πρέπει να καλέσουμε την συνάρτηση και να εκτυπώσουμε το αποτέλεσμα .Έστω ότι θέλουμε να υπολογίσουμε το εμβαδόν τριγώνου με βάση 3 και ύψος 8 τότε γράφουμε το όνομα της συνάρτησης και τις τιμές των παραμέτρων μας που στην περίπτωση μας είναι 3 και 8.

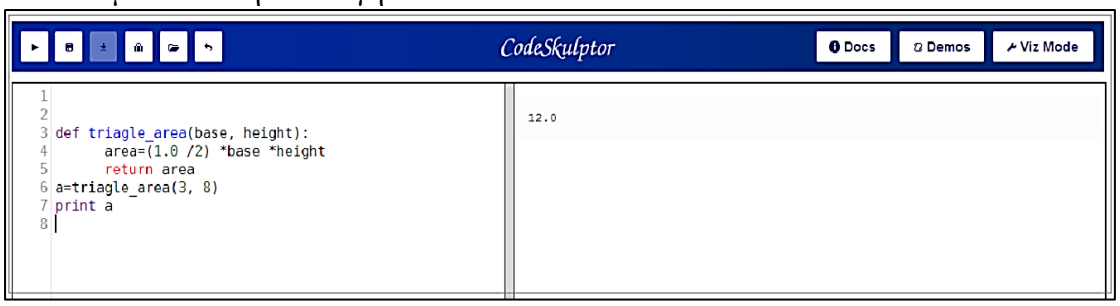
```
a=triangle_area (3, 8)
print a
```

Εάν τρέξουμε το πρόγραμμα η έξοδος του προγράμματος θα είναι 12.Οι συναρτήσεις μας είναι πολύ χρήσιμες αφού μπορούμε να αλλάξουμε τις παραμέτρους τις και να τις καλέσουμε ανά πάσα στιγμή μέσα στο πρόγραμμα .Δεν πρέπει να ξεχνάμε ότι το base και height υπάρχουν μόνο μέσα στην συνάρτηση ,δεν τα χρησιμοποιούμε ποτέ εκτός συνάρτησης.

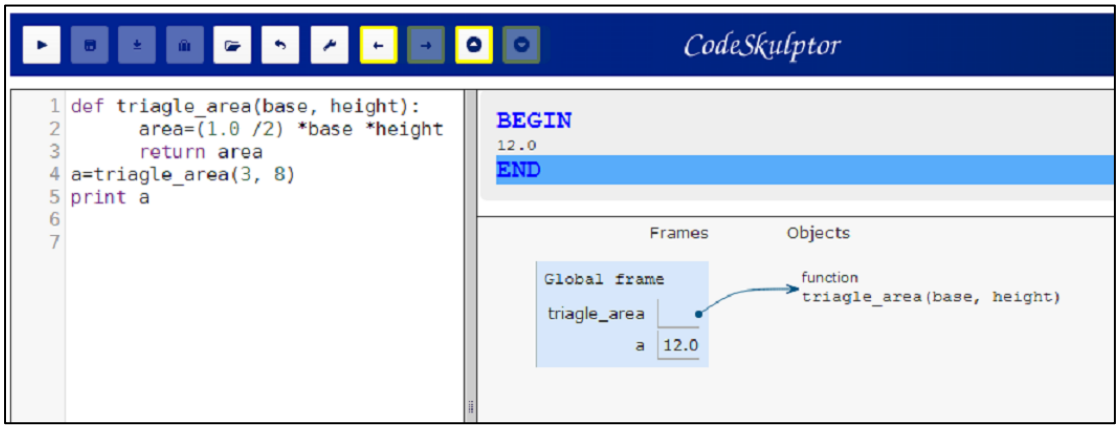
2.2 VISUALISING FUNCTIONS

Το VIZ MODE είναι μια λειτουργία του CODESKUPTOR το οποίο μας επιτρέπει να δούμε τι συμβαίνει όταν το CODESKUPTOR τρέχει ένα πρόγραμμα. Το VIZ MODE είναι ιδιαίτερα χρήσιμο για κάποιον καινούργιο προγραμματιστή αφενός μεν μπορούμε να κατανοήσουμε καλύτερα πως λειτουργεί η PYTHON αφετέρου δε μπορούμε να ανιχνεύσουμε και να εξαλείψουμε με μεγαλύτερη ευκολία σφάλματα του προγράμματος. Ας δούμε καλύτερα πως λειτουργεί το VIZ MODE με ένα παράδειγμα.

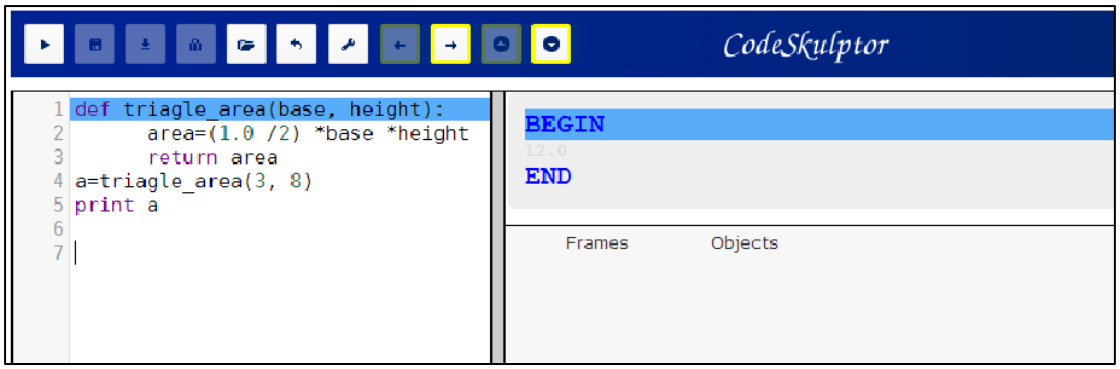
1.Κάνουμε κλικ στην επιλογή VIZ MODE



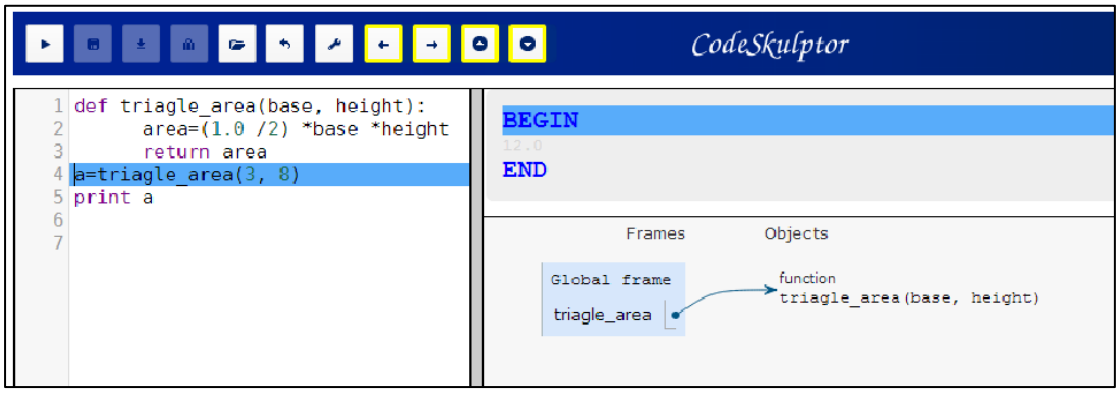
2.Το CODESKUPTOR ανοίγει ένα καινούργιο παράθυρο όπου αντιγράφουμε το πρόγραμμα μας. Κατόπιν το τρέχουμε. Όπως βλέπουμε και στην παρακάτω εικόνα το CODESKUPTOR τρέχει το πρόγραμμα αλλά τώρα καταγράφει όλα τα βήματα μέχρι την ολοκλήρωση του προγράμματος. Η μπλε μπάρα μας δείχνει σε ποιο σημείο του πρόγραμμα μας βρίσκεται. Για να εξετάσουμε την λειτουργία του προγράμματος μας αρκεί να μετακινήσουμε την μπλε μπάρα είτε με το αριστερό ή το δεξί βελάκι που βρίσκονται στην μπάρα



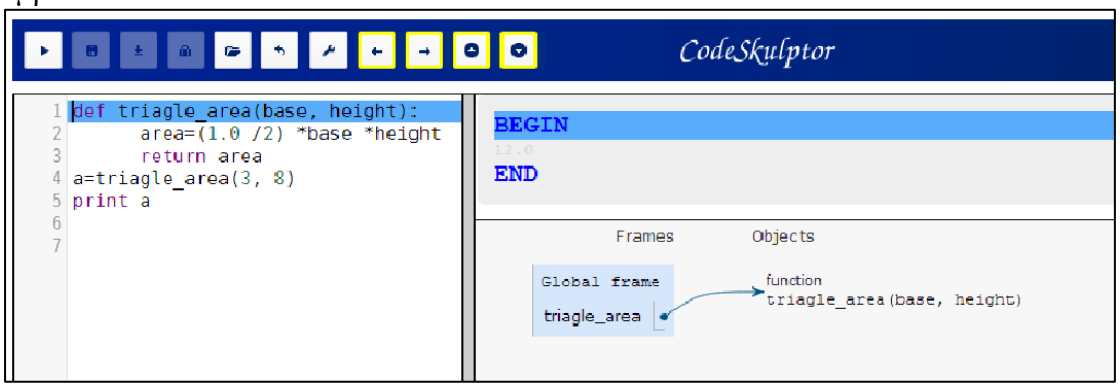
3.Η μπάρα βρίσκεται στην εκκίνηση του προγράμματος. Με το δεξί βελάκι μετακινούμε την μπάρα στο επόμενο βήμα.



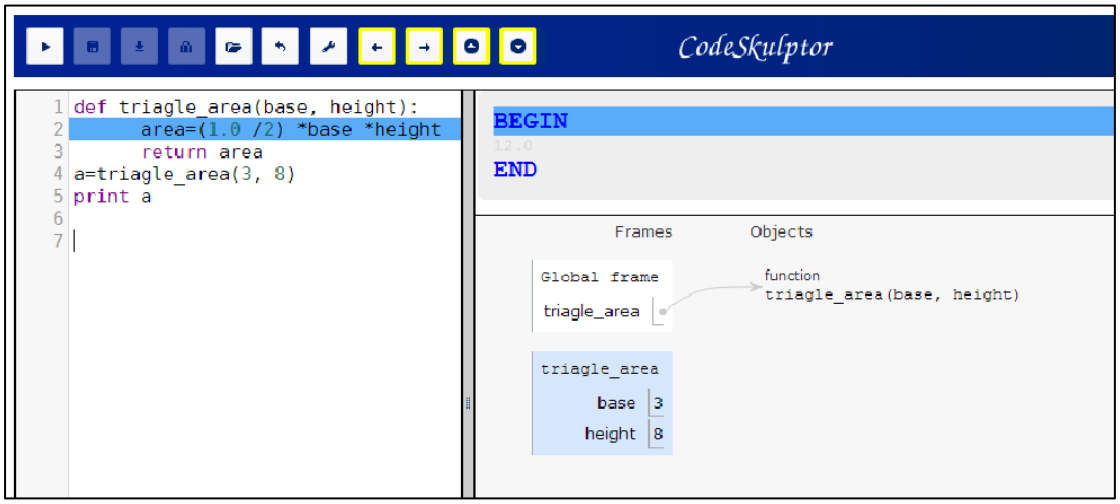
4. Το CODESKUPTOR διαβάζει την συνάρτηση triangle_area και τις παραμέτρους τις.



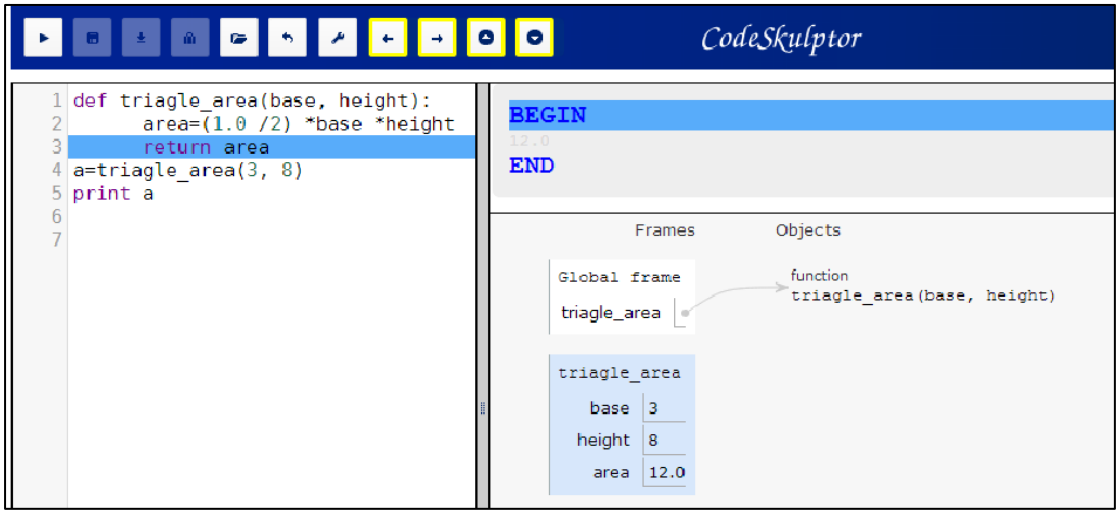
5. Στο επόμενο βήμα διαβάζει την συνάρτηση που triangle_area για να υπολογίσει το εμβαδόν



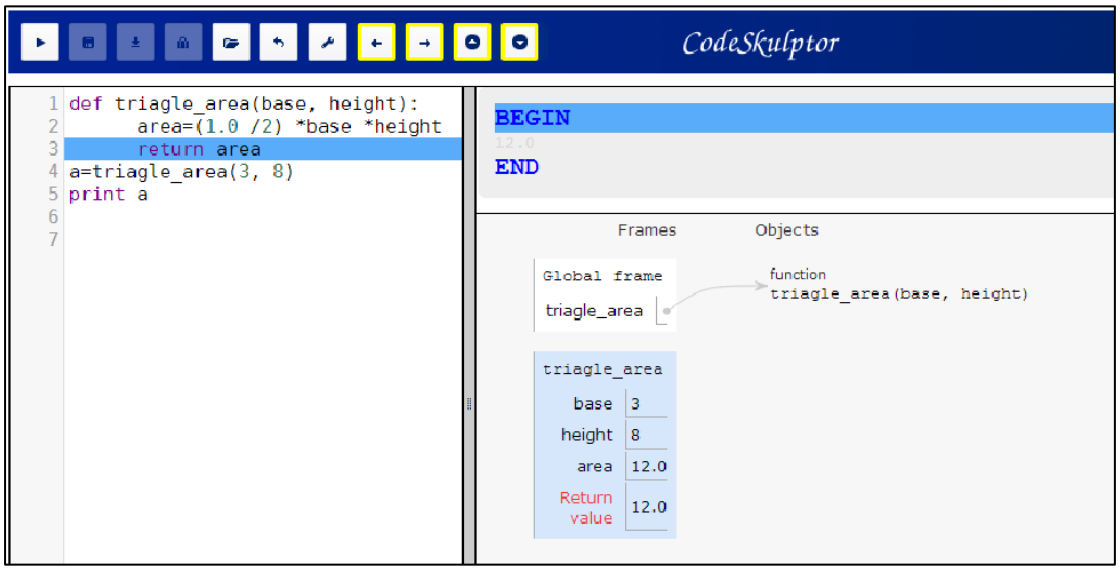
6.Το CODESKULPTOR αντικαθιστά το base και height με το 3 και 8 αντίστοιχα.



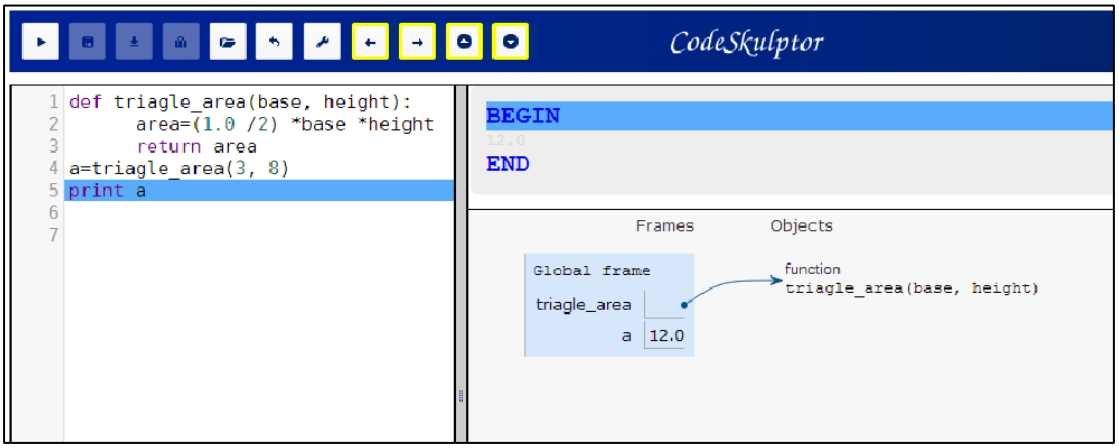
7. Υπολογίζεται το εμβαδόν area



8. Επιστρέφεται η τιμή area



9.Εκτυπώνεται η τιμή του a



2.3BOOLEANS (TRUE-FALSE)

Στην PYTHON όπως και σε όλες τις γλώσσες προγραμματισμού τα δεδομένα τύπου Boolean παριστάνονται με τιμές TRUE ή FALSE .Οι πράξεις BOOLEAN περιλαμβάνουν τρείς τελεστές NOT, OR και AND και χρησιμοποιούνται σε υποθετικές εκφράσεις if,while.Παρακάτω δίνεται ο πίνακας αληθείας για τους τελεστές NOT,OR,AND.

A	B	C	A AND B	A OR B	NOT C
F	F	F	F	F	F
F	T	T	F	T	T
T	F		F	T	
T	T		F	T	

Εικόνα 1:Πίνακας αληθείας

Τελεστές σύγκρισης στην PYTHON :

<

<

!=

==

<=

>=

Με τους παραπάνω τελεστές συγκρίνουμε δύο μεταβλητές μεταξύ τους και παράγουμε μία μεταβλητή τύπου Boolean.

Παραδείγματα σύγκρισης:

```
a=8>3

print a
```

Αφού το οκτώ είναι όντως μεγαλύτερο από το τρία τότε η έξοδος μας θα είναι TRUE.

Μπορούμε επίσης να συγκρίνουμε δεδομένα τύπου string.

```
A="hello"=='hello'

print A
```

Η έξοδος του προγράμματος μας θα είναι True.Ας μην ξεχνάμε ότι το = είναι ανάθεση τιμής στην μεταβλητή Α ενώ το == είναι ο τελεστής σύγκρισης.

2.4 RANDOM MODULE

Το random module περιέχει συναρτήσεις που εμπεριέχουν τυχαία αποτελέσματα. Για να γίνει δυνατή η χρησιμοποίηση των συγκεκριμένων συναρτήσεων πρέπει να δηλωθεί το σχετικό module που στην συγκεκριμένη περίπτωση είναι το random με την δήλωση import random.

1. random.choice (a_seq)

Παράδειγμα:

```
import random
print random.choice ([1, 2, 3, 4, 5, 6])
```

Εκτυπώνεται ένας τυχαίος αριθμός από την παραπάνω ακολουθία.

2. random.randint(start, stop)

Παράδειγμα:

```
import random
print random.randint(0, 10)
```

Εκτυπώνεται ένας αριθμός από το 0 έως το 10.

3. random.randrange(start, stop)

Παράδειγμα:

```
import random
print random.randrange(0, 10)
```

Εκτυπώνεται ένας αριθμός από το 0 έως το 9.

4. random.shuffle(a_list)

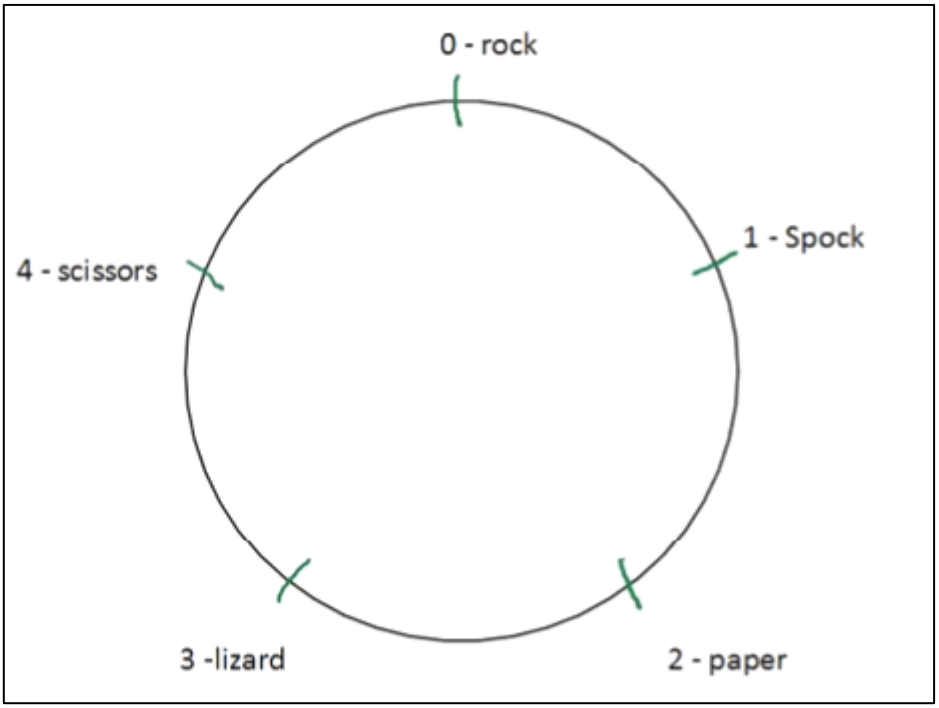
Παράδειγμα:

```
import random
numbers = range (5)
random.shuffle(numbers)
print numbers
```

Εκτυπώνεται η λίστα numbers η οποία περιέχει τους αριθμούς από 0 έως 4 σε τυχαία σειρά.

2.5 ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΑΣΚΗΣΗ

Την πρώτη εβδομάδα του μαθήματος υλοποιήθηκε το Rock-Paper-Scissors-Lizard-Spock.Το συγκεκριμένο παιχνίδι που δημιουργήθηκε από τον Sam Kass και την Karen Bryla είναι επέκταση του κλασσικού παιχνιδιού Rock-Paper-Scissors .Λειτουργεί με τις ίδιες αρχές του Rock-Paper-Scissors όμως προσθέτει δύο επιπλέον μεταβλητές(lizard /Spock).



Εικόνα 2:Πίνακας Rock-Paper-Scissors-Lizard-Spock

Από το παραπάνω σχήμα μπορούμε δούμε ποιος είναι ο νικητής. Κάθε παίκτης νικάει τον αντίπαλο που βρίσκεται στα αριστερά του και χάνει από τον αντίπαλο που βρίσκεται στα δεξιά του. Για παράδειγμα ο Spock νικάει τους αντιπάλους που βρίσκονται στα αριστερά του και χάνει από αυτούς που βρίσκονται στα δεξιά του. Ο καλύτερος τρόπος υλοποίησης τέτοιου είδους παιχνιδιών είναι το modular arithmetic. Εάν δούμε το παραπάνω σχήμα παρατηρούμε ότι κάθε επιλογή αντιστοιχίζεται σε έναν αριθμό. Έστω ότι κάθε παίκτης επιλέγει έναν αριθμό από το 0 έως το 4. Με το modular arithmetic αφαιρούμε την επιλογή του πρώτου παίκτη από την επιλογή του δεύτερου παίκτη και υπολογίζουμε το υπόλοιπο του αποτελέσματος με το 5.Ο πρώτος παίκτης είναι νικητής εάν το αποτέλεσμα είναι 1 ή 2 και ο δεύτερος παίκτης είναι νικητής εάν το αποτέλεσμα είναι 3 ή 4.Εάν το αποτέλεσμα είναι 0 τότε το παιχνίδι είναι ισοπαλία.

2.6 Διαδικασία ανάπτυξης προγράμματος

1.Σχεδιασμός συνάρτησης name_to_number(name) που αντιστοιχεί μια μεταβλητή τύπου string(name) που στην συγκεκριμένη περίπτωση είναι οι επιλογές «rock», «paper», «Spock», «lizard», «scissors» σε έναν αριθμό μεταξύ 0 έως 4 όπως περιγράφεται παραπάνω.

```
5  def name_to_number(name):
6      if name == 'rock':
7          return 0
8      elif name=='Spock':
9          return 1
10     elif name=='paper':
11         return 2
12     elif name=='lizard':
13         return 3
14     elif name=='scissors':
15         return 4
16     else:
17         return 'wrong input'
```

2.Σχεδιασμός συνάρτησης number_to_name(number) που αντιστοιχεί την μεταβλητή number εύρους 0 έως 4 σε μία μεταβλητή τύπου string.

```
19 def number_to_name(number):
20
21
22     if number == 0:
23         return 'rock'
24     elif number==1:
25         return 'Spock'
26     elif number==2:
27         return 'paper'
28     elif number==3:
29         return 'lizard'
30     elif number==4:
31         return 'scissors'
32     else:
33         return 'wrong input'
```

3.Σχεδιασμος του πρώτου μέρους της κύριας συνάρτησης rpsls(guess)που μετατρέπει την επιλογή του χρήστη(player_choice) σε έναν αριθμό(player_number) μεταξύ 0-4 χρησιμοποιώντας την συνάρτηση name_to_number(name).

```
35
36 def rpsls(player_choice):
37     player_number=name_to_number(player_choice)
38
```

4. Σχεδιασμός του δεύτερου μέρους της κύριας συνάρτησης rpsls(guess) που δημιουργεί έναν τυχαίο αριθμό(comp_number) από το 0-4 χρησιμοποιώντας την συνάρτηση random.randrange()

```
37
38     comp_number=random.randrange(0, 5)
39
```

5.Σχεδιασμός του τελευταίου μέρους της κύριας συνάρτησης rpsls(guess) που αποφασίζει και εκτυπώνει ποιος είναι ο νικητής. Παίρνουμε την διαφορά (comp_number-player_number) και εφαρμόζουμε modular arithmetic(%) στο αποτέλεσμα.

```
41     print""
42     print"Player chooses",player_choice
43     print "Computer chooses",comp_name
44     difference=(player_number-comp_number)%5
45     if difference==0:
46        print 'it is a tie!'
47     elif difference==1:
48          print 'player wins!'
49     elif difference==2:
50          print 'player wins!'
51     elif difference==3:
52          print 'computer wins!'
53     elif difference==4:
54          print 'computer wins!'
55
```

7.Καλούμε την rpsls με τις 5 επιλογές ξεχωριστά.

```
58 # test your code
59 rpsls("rock")
60 rpsls("Spock")
61 rpsls("paper")
62 rpsls("lizard")
63 rpsls("scissors")
64
```

ΕΞΟΔΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

```
Player chooses rock
Computer chooses paper
computer wins!

Player chooses Spock
Computer chooses Spock
it is a tie!

Player chooses paper
Computer chooses scissors
computer wins!

Player chooses lizard
Computer chooses paper
player wins!

Player chooses scissors
Computer chooses scissors
it is a tie!
```

Όπως παρατηρούμε στο βήμα 7 καλούμε την συνάρτηση rpsls πέντε φορές η οποία παίρνει σαν παράμετρο την επιλογή του παίκτη. Η επιλογή του παίκτη παραμένει πάντα η ίδια όσες φορές και εάν τρέξουμε το πρόγραμμα .Το μόνο που αλλάζει είναι η επιλογή του δεύτερου παίκτη με την βοήθεια της συνάρτησης random.randrange που στην περίπτωση μας είναι η επιλογή του υπολογιστή(comp_number).

3. Guess the number

Την δεύτερη εβδομάδα του μαθήματος υλοποιήθηκε το Guess the number . Για τις απαιτήσεις της συγκεκριμένης άσκησης απαιτείται η εξοικείωση του αναγνώστη με τον προγραμματισμό με συμβάντα(event driven programming). Αρχικά αναλύεται η έννοια προγραμματισμός με συμβάντα και παρατίθενται παραδείγματα προγραμματισμού με συμβάντα. Επιπροσθέτως παρουσιάζεται ο τρόπος δημιουργίας ενός frame και οι περιοχές ελέγχου του frame που απαιτούνται για την δημιουργία διαδραστικών εφαρμογών.

3.1 Local vs global μεταβλητές

Μεταβλητές οι οποίες ορίζονται έξω από μία συνάρτηση τις οποίες μπορούμε να τροποποιήσουμε ονομάζουμε global μεταβλητές ενώ οι μεταβλητές που ορίζονται μέσα σε μια συνάρτηση τις οποίες δεν μπορούμε να τροποποιήσουμε εκτός συνάρτησης τις ονομάζουμε τοπικές μεταβλητές .Στο παρακάτω πρόγραμμα το num1 είναι ένα παράδειγμα global μεταβλητής ενώ το num2 που βρίσκεται στην συνάρτηση fun() είναι τοπική μεταβλητή.

```
1
2  num1 = 1
3  print num1
4
5  def fun():
6      num2 = 2
7      print num2
8
9
10 fun()
11 print num1
12
```

```
1
2
1
```

Η έξοδος του προγράμματος είναι η αναμενόμενη αλλά τι θα γίνει άμα προσθέσουμε ένα «print num2» στο πρόγραμμα μας .

```
1
2  num1 = 1
3  print num1
4
5  def fun():
6      num2 = 2
7      print num2
8
9
10 fun()
11 print num1
12 print num2
13
```

```
1
2
1

Line 12: NameError: name 'num2' is not defined
```

Το CodeSkulptor εμφάνισε «λάθος όνομα το num2 δεν έχει ορισθεί». Πολύ λογικό αφού το num2 είναι τοπική μεταβλητή και μπορεί να τροποποιηθεί και να

χρησιμοποιηθεί μόνο εντός της fun().Ας δούμε πώς μπορούμε να χρησιμοποιήσουμε την num2 εκτός συνάρτησης κάνοντας την global. Στον παραπάνω κώδικα το num2 εκτυπώθηκε χωρίς να καλέσουμε την συνάρτηση fun().Επειδή δηλώσαμε την num2 σαν global μεταβλητή μπορούμε να την χρησιμοποιήσουμε οπουδήποτε στο πρόγραμμα.

3.2 Event driven programming(Προγραμματισμός με συμβάντα)

Ο προγραμματισμός με συμβάντα είναι ένα παράδειγμα προγραμματισμού όπου η ροή του προγράμματος σταματάει μέχρι να συμβούν διάφορα γεγονότα (events). Ο προγραμματισμός με συμβάντα μας επιτρέπει να σχεδιάζουμε διαδραστικές εφαρμογές όπως γραφικά περιβάλλοντα αλληλεπίδρασης χρήστη-υπολογιστή.

Παραδείγματα βασικών συμβάντων(events) στην PYTHON:

1.Input(είσοδος)

• Button

• Text box

2. Keyboard (πληκτρολόγιο)

•Key down

• Key up

3. Mouse (ποντίκι)

• Click

• Drag

4.Timer

3.3 Παράδειγμα ενός απλού προγράμματος event driven

1.Πρώτα πρέπει να δηλωθεί το simpleGUI. Το simpleGUI είναι ένα γραφικό περιβάλλον χρήστη της PYTHON που μας επιτρέπει να σχεδιάζουμε διαδραστικές εφαρμογές.

```
import simplegui
```

2.Ορίζουμε το event handler δηλαδή ένα κομμάτι κώδικα που διαχειρίζεται ένα συμβάν. Στην περίπτωση μας είναι μια συνάρτηση χωρίς ορίσματα που το μόνο που κάνει είναι να εκτυπώνει την λέξη ‘’tick’’.

```
def tick ():
    print 'tick'
```

3.Δηλώνουμε το event handler .Με την βοήθεια του simpleGUI καλείται η συνάρτηση create_timer () η οποία δημιουργεί ένα χρονόμετρο. Η συνάρτηση δέχεται

δύο ορίσματα τον χρόνο σε milliseconds που θέλουμε να επαναλαμβάνεται το συμβάν και το event handler που ορίσαμε παραπάνω.

```
timer = simplegui.create_timer (1000, tick)
```

4.Καλούμε την συνάρτηση timer.start() που ξεκινάει το χρονόμετρο.

```
timer.start ()
```

ΕΞΟΔΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

```
tick
tick
tick
tick
tick
tick
tick
tick
tick
tick
tick
tick
tick
tick
tick
tick
tick
tick
tick
tick
tick
tick
tick
```

3.4 Δημιουργία frame και περιοχές ελέγχου frame

Ένα frame είναι ένα παράθυρο στο οποίο απεικονίζονται τα αντικείμενα ελέγχου και τα σχήματα του προγράμματος..

ΣΥΝΤΑΞΗ:

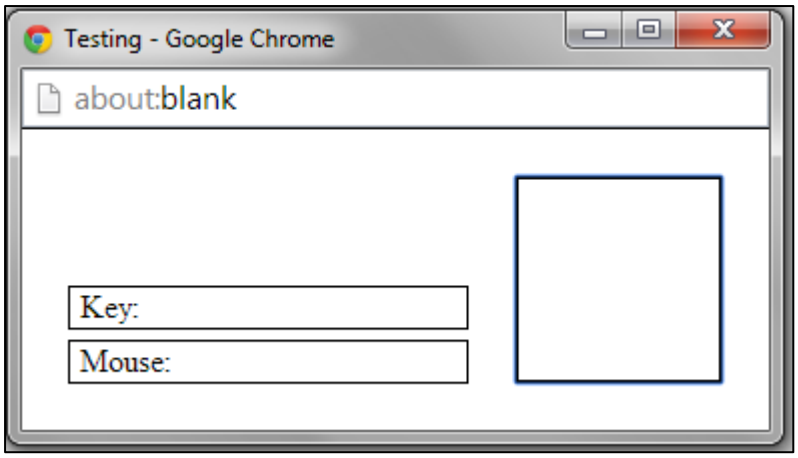
```
frame = simplegui.create_frame (title, canvas_width, canvas_height)
```

Παράδειγμα:

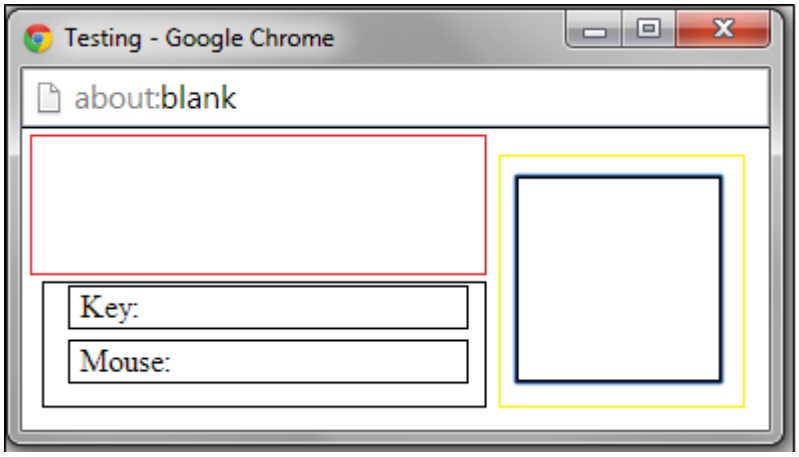
```
import simplegui

frame = simplegui.create_frame('Testing', 100, 100)
```

ΕΞΟΔΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

Δημιουργία frame με όνομα 'Testing' με πλάτος και ύψος 100x100 pixels



Περιοχές ελέγχου του frame:



Περιοχή ελέγχου (control area): Το κόκκινο πλαίσιο στο frame ονομάζεται περιοχή ελέγχου του frame.Στην περιοχή ελέγχου του frame απεικονίζονται τα buttons και τα πεδία κειμένου(input text).

Περιοχή του καμβά(canvas area): Η περιοχή στο κίτρινο πλαίσιο του frame ονομάζεται καμβάς. Στον καμβά απεικονίζονται τα μηνύματα, τα σχέδια και οι εικόνες του προγράμματος.

Περιοχή κατάστασης(status area):Στην περιοχή κατάστασης απεικονίζεται η λειτουργία του πληκτρολογίου και του ποντικού που χρησιμοποιούνται από τον χρήστη για τον έλεγχο του προγράμματος.

3.5 Δημιουργία button

Ένα button είναι ένα αντικείμενο ελέγχου που τοποθετείται στην περιοχή ελέγχου(control area) του frame.

ΣΥΝΤΑΞΗ:

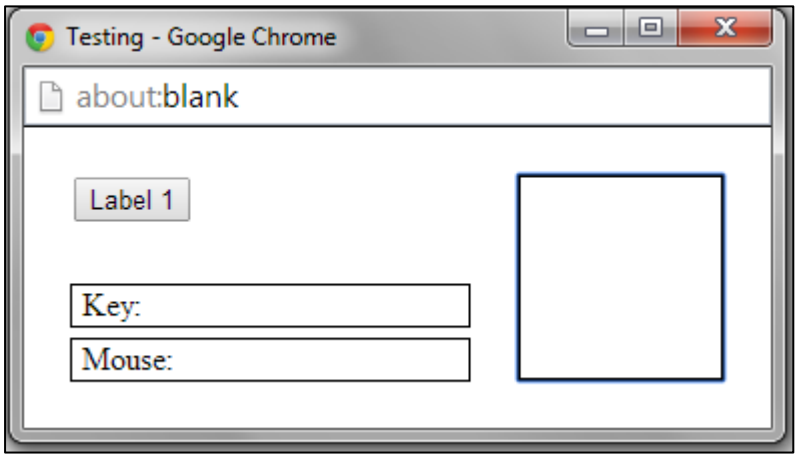
frame.add_button(text, button_handler)

Παράδειγμα:

```
import simplegui

def button_handler ():

    pass

frame = simplegui.create_frame('Testing', 100, 100)

button = frame.add_button('Label 1', button_handler)
```

ΕΞΟΔΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

Δημιουργία button στην περιοχή control area με όνομα ‘Label 1’

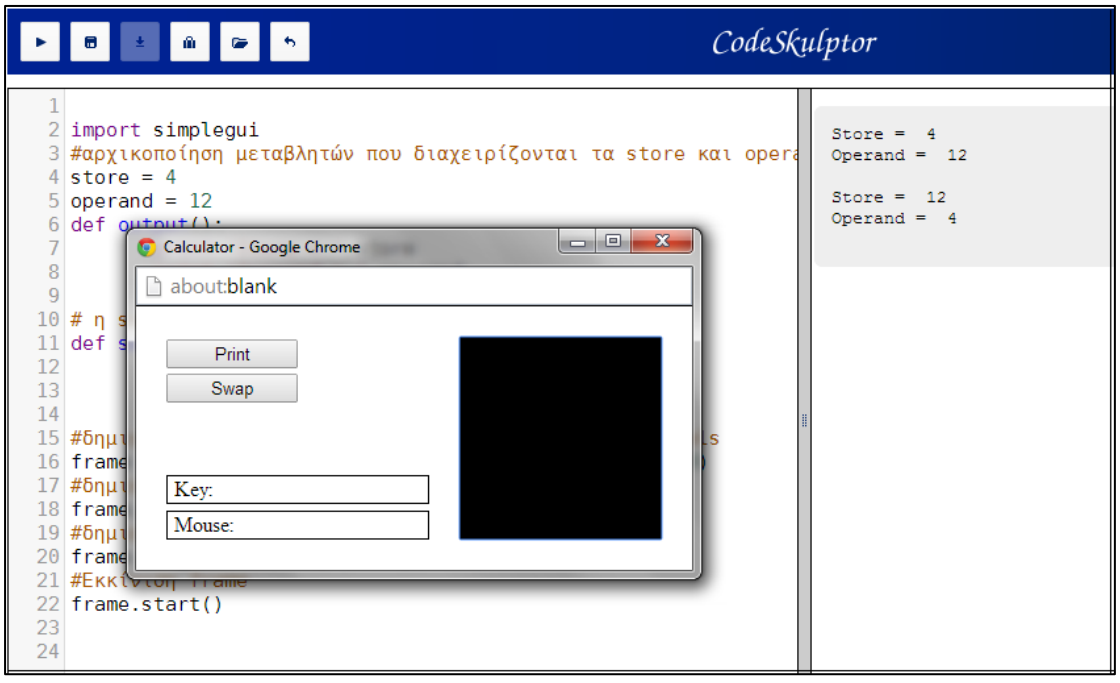


3.5.1 Παράδειγμα διαχείρισης button

```
import simplegui
Αρχικοποίηση μεταβλητών που διαχειρίζονται τις μεταβλητές store και operand
store = 4
operand = 12

Δημιουργία συνάρτησης output η οποία κάθε φορά που καλείται τυπώνει τις τιμές των
store και operand
def output():
      print “Store = ”,store
      print “Operand = ”,operand
      print””
```

```
Η swap αντιμεταθέτει τις τιμές store και operand
def swap():
    global store, operand
    store, operand = operand, store
    output()
Δημιουργία frame με το όνομα Calculator 150x150 pixels
frame = simplegui.create_frame("Calculator", 150, 150)

Δημιουργία button με το όνομα print στην περιοχή ελέγχου του frame
Σαν button_handler του  button "Print" ορίζεται η συνάρτηση output
frame.add_button("Print", output, 100)

Δημιουργία button με το όνομα swap
Σαν button_handler του  button "swap" ορίζεται η συνάρτηση swap
 frame.add_button("Swap", swap, 100)
Εκκίνηση frame
frame.start()
```

ΕΞΟΔΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ



Όπως παρατηρούμε στο frame δημιουργήθηκαν δύο buttons print και swap. Ο χρήστης κάθε φορά που κάνει κλικ στο button «Print» καλείται το button_handler του button δηλαδή η συνάρτηση output η οποία εκτυπώνει στην οθόνη τις τιμές του store και του operand αντίστοιχα το ίδιο συμβαίνει με το button swap το οποίο κάνει αντιμετάθεση των τιμών του store και operand και τα εμφανίζει στην οθόνη.

3.6 Δημιουργία πεδίου κειμένου

Ένα πεδίο κειμένου είναι ένα αντικείμενο ελέγχου το οποίο δέχεται μία παράμετρο τύπου string.

ΣΥΝΤΑΞΗ:

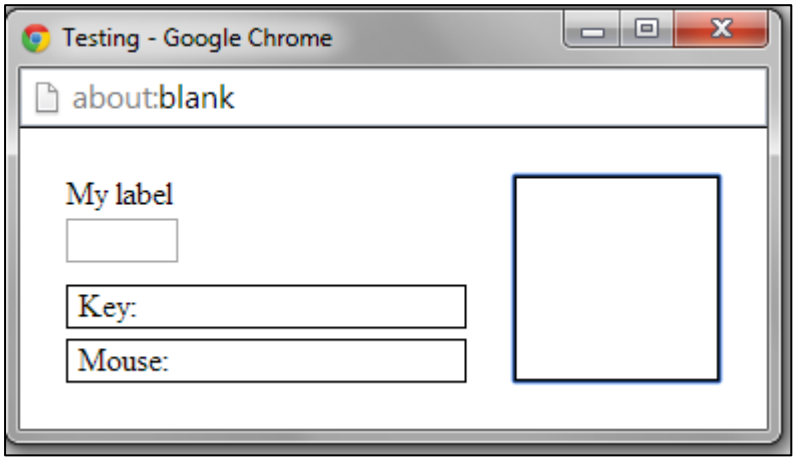
```
frame.add_input(text, input_handler, width)
```

Παράδειγμα:

```
import simplegui

def input_handler ():

 pass

frame = simplegui.create_frame('Testing', 100, 100)

inp = frame.add_input('My label', input_handler, 50)
```

ΕΞΟΔΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

Δημιουργία πεδίου κειμένου 'My label'στην περιοχή control area όπου ο χρήστης εισάγει δεδομένα.

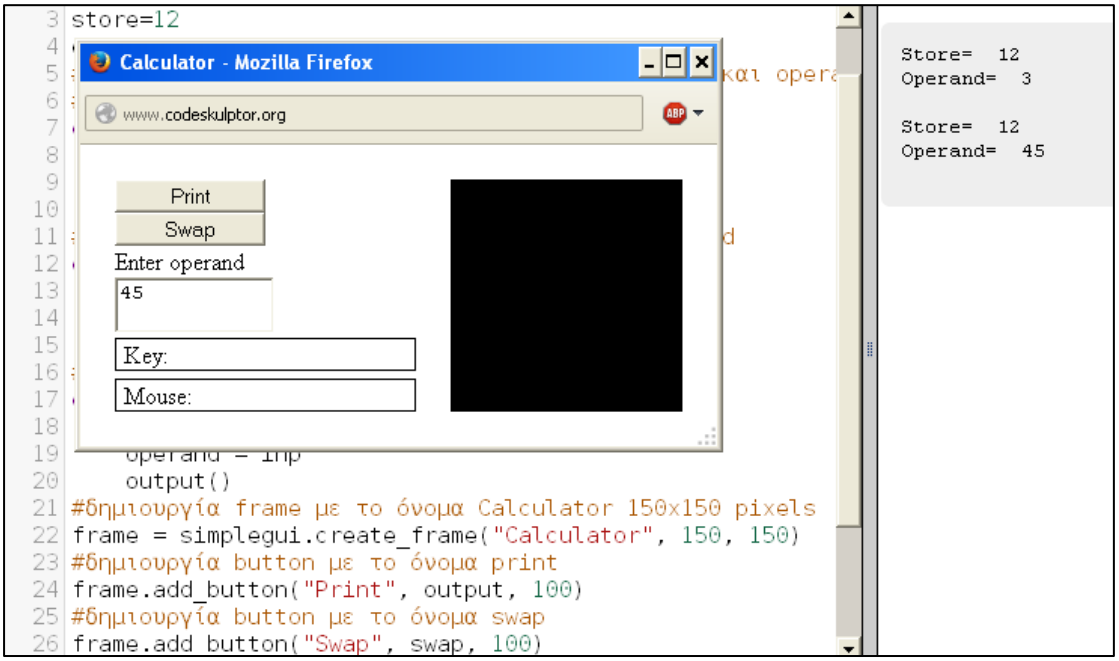


3.6.1 Παράδειγμα διαχείρισης πεδίου κειμένου

```
import simplegui
Αρχικοποίηση μεταβλητών που διαχειρίζονται τα store και operand
store = 4
operand = 12

Δημιουργία συνάρτησης output η οποία εκτυπώνει τις τιμές των store και operand
def output():
    print “Store = ”,store
    print “Operand = ”,operand
    print””
```

```
Δημιουργία συνάρτησης swap η οποία αντιμεταθέτει τις τιμές store και operand
Καλείται η output η οποία εκτυπώνει τις τιμές των store και operand
def swap():
    global store, operand
    store, operand = operand, store
    output()
Η συνάρτηση enter εισάγει έναν αριθμό στην μεταβλητή operand
def enter(inp):
global store, operand
operand=inp
output()
Δημιουργία frame με το όνομα Calculator 150x150 pixels
frame = simplegui.create_frame("Calculator", 150, 150)
Δημιουργία  button με το όνομα print το οποίο καλεί την συνάρτηση output
frame.add_button("Print", output, 100)
Δημιουργία button με το όνομα swap το οποίο καλεί την συνάρτηση swap
frame.add_button("Swap", swap, 100)

Δημιουργία πεδίου κειμένου Enter operand στο οποίο ορίζεται σαν «input handler» η συνάρτηση enter
frame.add_button("Enter operand", enter, 100)
Εκκίνηση frame
frame.start()
```

ΕΞΟΔΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ



Όπως βλέπουμε παραπάνω στο frame δημιουργήθηκε το πεδίο κειμένου “Enter operand” 'όπου ο χρήστης εισάγει έναν αριθμό και αλλάζει την τιμή του operand από 3 σε 45.

3.7 ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΑΣΚΗΣΗ

Στην δεύτερη εβδομάδα του μαθήματος υλοποιήθηκε το Guess the number.Το παιχνίδι υλοποιείται από δύο παίκτες. Ο πρώτος παίκτης που στην περίπτωση μας είναι ο υπολογιστής επιλέγει έναν αριθμό από το 0 έως το 100 τον οποίο κρατάει κρυφό από τον δεύτερο παίκτη.Ο δεύτερος παίκτης προσπαθεί να μαντέψει ποιος είναι αυτός ο κρυφός αριθμός.Πιο συγκεκριμένα πρέπει να δημιουργηθούν στο frame δύο buttons ένα για εύρος από 0 έως 100 και ένα για εύρος από 0 έως 1000.Ο δεύτερος παίκτης θα έχει την δυνατότητα να επιλέξει το εύρος επιλογής αριθμού κάνοντας κλικ σε ένα από τα δύο buttons.Επίσης πρέπει να δημιουργηθεί ένα πεδίο κειμένου όπου ο χρήστης εισάγει τον αριθμό που πιστεύει ότι έχει επιλέξει ο πρώτος παίκτης.Αναλογα με το εύρος επιλογής ο δεύτερος παίκτης θα έχει συγκεκριμένο αριθμό ευκαιριών να μαντέψει τον κρυφό αριθμό.Εάν ο παίκτης μαντέψει τον κρυφό αριθμό προτού τελειώσουν οι ευκαιρίες του τότε το πρόγραμμα εμφανίζει το μήνυμα ΄Κέρδισες΄ αλλιώς εμφανίζει ΄Εχασες΄.

3.7.1 ΥΛΟΠΟΙΗΣΗ ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗΣ ΑΣΚΗΣΗΣ

```
Δήλωση βιβλιοθηκών simplegui και random
import simplegui
import random


Αρχικοποίηση μεταβλητών count  και secret_num
secret_num = 0;
count = 0;



Στην εκκίνηση του παιχνιδιού το εύρος κυμαίνεται από 0-100
Δίνεται τιμή True στην μεταβλητή game_100
game_100 = True


Βοηθητική συνάρτηση  new_game που εκκινεί το πρόγραμμα ανάλογα με το εύρος του παιχνιδιού
Όταν η τιμή του game100 είναι True το εύρος του παιχνιδιού κυμαίνεται από 0-100
αλλιώς όταν η τιμή του η game100 είναι False το εύρος του παιχνιδιού κυμαίνεται από 0-1000
def new_game():
  global secret_num, count
  if game_100:
```

```
        range100()
    else:
        range1000()
```

Ορισμός event handler για εύρος 0-100
Η random.randint παράγει έναν ακέραιο αριθμό απο το 0 έως 100
Η μεταβλητή count δηλώνει τον αριθμό των ευκαιριών που έχει κάθε παίχτης για να μαντέψει τον κρυφό αριθμό.
Κάθε φορά που καλείται η συνάρτηση range100 εκτυπώνεται στην οθόνη το εύρος του παιχνιδιού και ο αριθμός των ευκαιριών που έχει ο παίκτης στην διάθεση του να μαντέψει τον μυστικό αριθμό.
Στην μεταβλητή secret_num εκχωρείται ένας τυχαίος ακέραιος αριθμός από το 0-100 με την βοήθεια της συνάρτησης random.randint

```
def range100():
    global secret_num, count, game_100
    game_100 = True
    secret_num = random.randint(0,100)
    count = 7
    print""
    print "New game. Range is from 0 to 100"
    print "Number of remaining guesses is", count
```

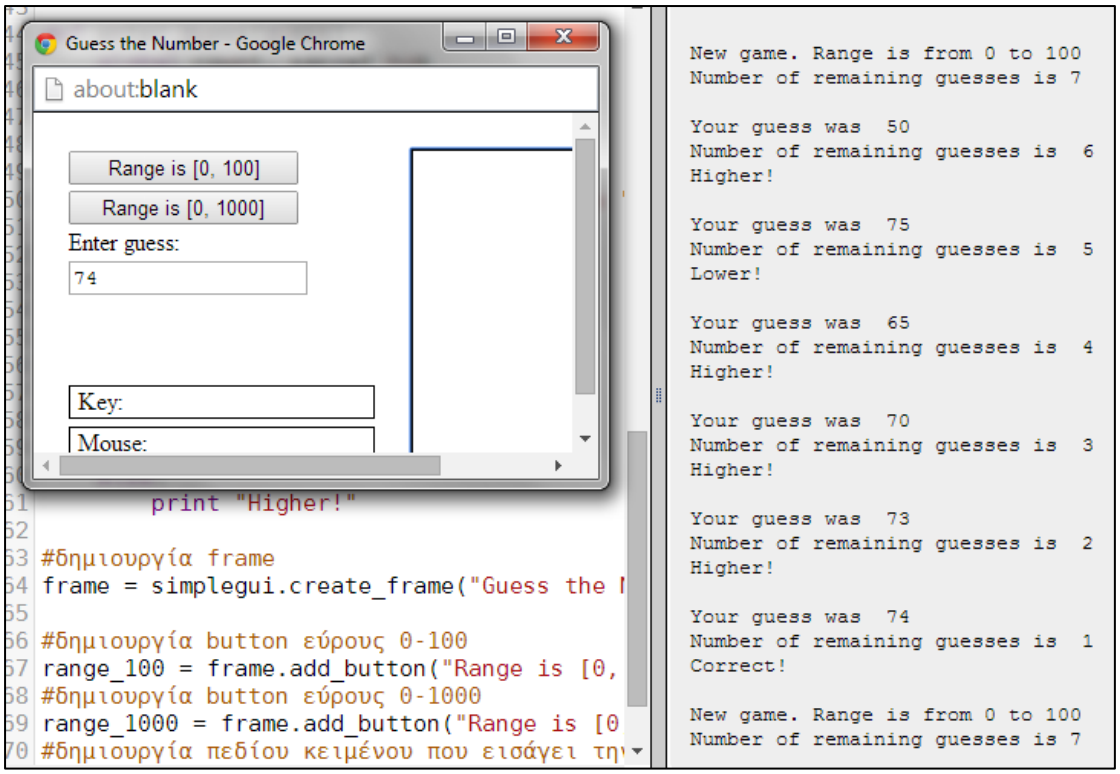
Ορισμός event handler για εύρος 0-1000
Η random.randint παράγει έναν ακέραιο αριθμό απο το 0 έως 1000
Η μεταβλητή count δηλώνει τον αριθμό των ευκαιριών που έχει κάθε παίχτης για να μαντέψει τον κρυφό αριθμό.
Στην μεταβλητή secret_num εκχωρείται ένας τυχαίος ακέραιος αριθμός από το 0-100 με την βοήθεια της συνάρτησης random.randint
Κάθε φορά που καλείται η συνάρτηση range1000 εκτυπώνεται στην οθόνη το εύρος του παιχνιδιού και ο αριθμός των ευκαιριών που έχει ο παίκτης στην διάθεση του να μαντέψει τον μυστικό αριθμό

```
def range1000():
    global secret_num, count, game_100
    game_100 = False
    secret_num = random.randint(0,1000)
    count = 10
    print""
    print "New game. Range is from 0 to 1000"
    print "Number of remaining guesses is", count
```

Δημιουργία συνάρτησης enter_guess που δέχεται σαν παράμετρο την επιλογή του χρήστη
Γίνεται σύγκριση της επιλογής του χρήστη με το μυστικό αριθμό(secret_num) που δημιουργήθηκε από την συνάρτηση random.randint των συναρτήσεων range100 και range1000 και εκτυπώνονται στην οθόνη τα ανάλογα μυνύματα.
Η μεταβλητή count μειώνεται κατά ένα κάθε φορά που ο παίκτης δίνει την επιλογή του

```
στο πεδίο κειμένου.
def enter_guess(guess):
    global count, secret_num
    count=count-1
    guess = int(guess)
    print""
    print "Your guess was ", guess
    print "Number of remaining guesses is ", count
    if count==0 and guess != secret_num:
       print "You run out of guesses"
       print ""
       new_game()
    elif guess == secret_num:
        print "Correct!"
        new_game()
    elif guess > secret_num:
        print "Lower!"
    else:
  print "Higher!"

Δημιουργία frame "Guess the Number",200x200 pixels
frame = simplegui.create_frame("Guess the Number", 200, 200)

Δημιουργία button εύρους 0-100
Όταν ο παίκτης κάνει κλικ στο button range100 καλείται η συνάρτηση range100 η
οποία ορίζει το εύρος επιλογής του παίκτη από 0 έως 100
range_100 = frame.add_button("Range is [0, 100]", range100, 150)


Δημιουργία button εύρους 0-1000
Όταν ο παίκτης κάνει κλικ στο button range1000 καλείται η συνάρτηση range1000 η
οποία ορίζει το εύρος επιλογής του παίκτη από 0 έως 1000
range_1000 = frame.add_button("Range is [0, 1000]", range1000, 150)

Δημιουργία πεδίου κειμένου που εισάγει την επιλογή του χρήστη
Το πεδίου κειμένου καλεί την συνάρτηση enter_guess η οποία δέχεται την επιλογή του
χρήστη και την συγκρίνει με τον τυχαίο αριθμό που έχει παραχθεί από την συνάρτηση
random.randint
input_guess = frame.add_input("Enter guess:", enter_guess, 150)

Καλείται η συνάρτηση new_game() που εκκινεί το παιχνίδι
new_game()
```

ΕΞΟΔΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ



4.STOPWATCH

Στο κεφάλαιο αυτό παρουσιάζουμε την υλοποίηση της προγραμματιστικής άσκησης Stopwatch.Για τις ανάγκες της προγραμματιστικής άσκησης απαιτείται η δημιουργία ενός χρονομέτρου στον καμβά. Στο συγκεκριμένο κεφάλαιο παρουσιάζονται οι εντολές σχεδίασης του Codeskulptor για την σχεδίαση αντικειμένων στον καμβά .

4.1 ΣΧΕΔΙΑΖΟΝΤΑΣ ΣΤΟ CODESKULPTOR

Το Codeskulptor προσφέρει μεγάλη γκάμα εντολών σχεδίασης όπως κείμενα,σχήματα και εικόνες .Για να σχεδιάσουμε στο Codeskulptor πρέπει να δηλωθεί το draw_handler .Το draw handler αντιπροσωπεύει μία συνάρτηση στο πρόγραμμα η οποία είναι υπεύθυνη για τον σχεδιασμό κάποιας εικόνας, σχήματος ή κειμένου τα οποία θέλουμε να εμφανίσουμε στον καμβά. Αφού καλέσουμε την συνάρτηση draw_handler το περιεχόμενο του καμβά ανανεώνεται αυτόματα. Ας δούμε αναλυτικά πώς χρησιμοποιούνται στην πράξη.

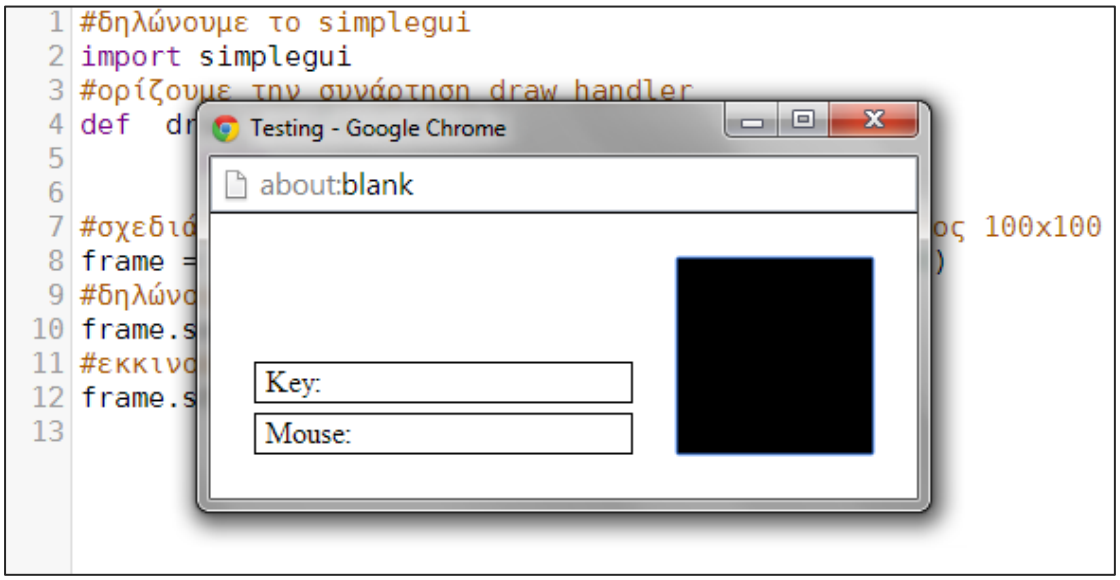
ΣΥΝΤΑΞΗ:

```
frame.set_draw_handler(draw_handler)
```

Παράδειγμα:

```
# Δήλωση βιβλιοθήκης simplegui
import simplegui

# Ορίζουμε την συνάρτηση draw handler την οποία καλεί το draw handler
# Στην παρακάτω συνάρτηση με την εντολή pass το Codeskulptor αγνοεί την συνάρτηση
# draw_handler μέχρις όπου δοθεί κάποια εντολή σχεδιασμού κειμένου ή σχήματος
    def draw_handler(canvas):
        pass

# Σχεδιασμός frame με όνομα "Testing" και μέγεθος 100x100 pixels
frame = simplegui.create_frame('Testing', 100, 100)

# Δήλωση του draw handler το οποίο καλεί την συνάρτηση draw_handler
frame.set_draw_handler(draw_handler)

# Εκκίνηση frame
frame.start()
```

ΕΞΟΔΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ



Όπως παρατηρούμε η έξοδος του προγράμματος είναι ένα κενό frame.Για να σχεδιάσουμε σχήματα ή κείμενο πρέπει δοθεί κάποια εντολή σχεδίασης στην συνάρτηση draw_handler.

4.1.1 ΣΧΕΔΙΑΖΟΝΤΑΣ ΚΕΙΜΕΝΟ

ΣΥΝΤΑΞΗ:

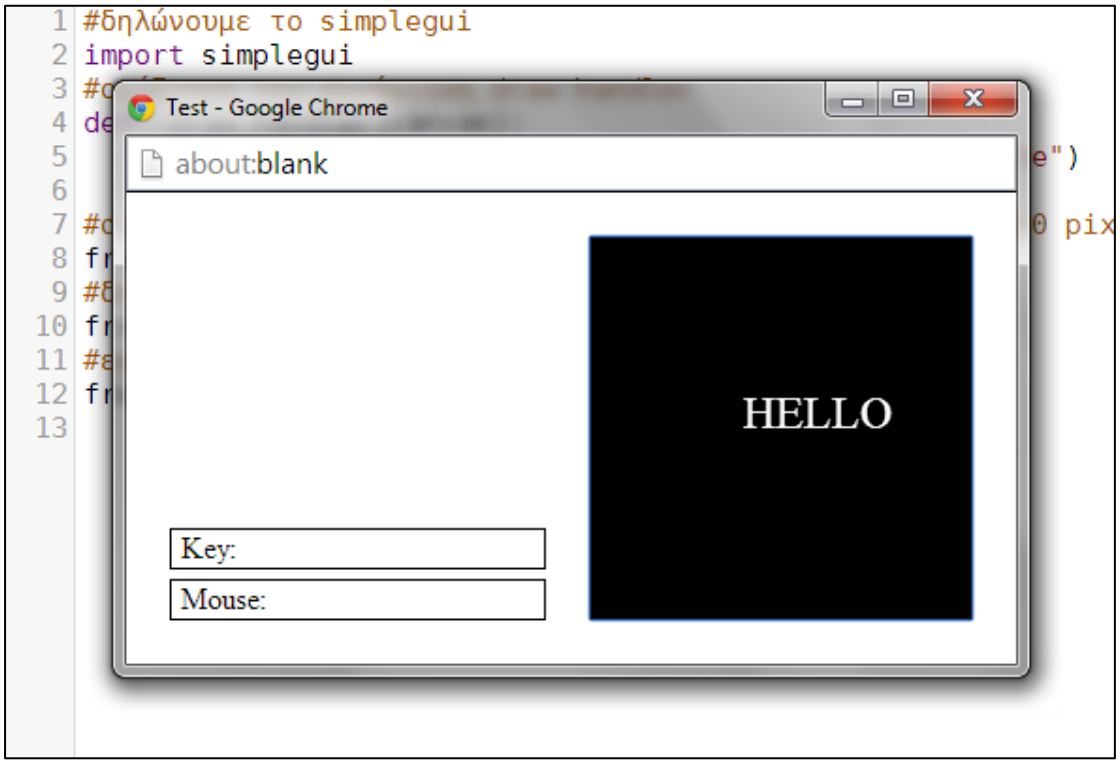
canvas.draw_text(“text”, point, font_size, font_color)

Παράδειγμα σχεδίασης του μηνύματος «HELLO» στον καμβά :

```
Δήλωση του simplegui
import simplegui

Ορίζουμε την συνάρτηση draw handler όπου με την εντολή canvas.draw_text
σχεδιάζουμε την λέξη «HELLO»
Το μήνυμα «HELLO» σχηματίζεται στο σημείο με συντεταγμένες[80,100] του καμβά
με μέγεθος 24 pixels και χρώμα λευκό
def draw_handler(canvas):
    canvas.draw_text(“HELLO”, [80, 100], 24, “White”)

Σχεδιάζουμε το frame με όνομα “Test” και μέγεθος 300x200 pixels
frame = simplegui.create_frame('Test', 100, 100)

Δήλωση του draw handler με το οποίο καλείται η συνάρτηση draw_handler και
σχηματίζεται στον καμβά η λέξη “HELLO”
```

```
frame.set_draw_handler(draw_handler)

Εκκίνηση του frame
frame.start()
```

ΕΞΟΔΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ



4.2 TIMERS

Δημιουργία χρονομέτρου

ΣΥΝΤΑΞΗ:

```
timer = simplegui.create_timer(time ,timer_handler)
```

Με την συγκεκριμένη εντολή το Codeskulptor δημιουργεί ένα χρονόμετρο το οποίο καλεί επανειλημμένα το event handler δηλαδή την συνάρτηση timer_handler στο χρόνο(time) σε milliseconds που έχουμε ορίσει.Το παρακάτω πρόγραμμα εμφανίζει στην οθόνη το μήνυμα “Python is fun”. Το χρονόμετρο καλεί το event handler δηλαδή την συνάρτηση tick η οποία μεταβάλλει την θέση του μηνύματος στον καμβά κάθε 2000 milliseconds.

```
Δήλωση βιβλιοθηκών import και random
import simplegui
import random
```

Δήλωση global μεταβλητών
Η θέση(position) του μηνύματος στην εκκίνηση του προγράμματος ορίζεται στην θέση (50x50) του καμβά

```
message=”Python is fun!”
position= [50, 50]
width = 500
height= 500
time_in_milliseconds= 2000
```

Ορισμός του event handler
Η random.radrange δίνει τυχαίες τιμές στο x και το y που είναι οι συντεταγμένες για την θέση του message στον καμβά.
Οι τιμές του x και y εκχωρούνται στην μεταβλητή position

```
def tick():
    x = random.randrange(0, width)
    y = random.randrange(0, height)
    position[0] = x
    position[1] = y
```

Ορισμός της συνάρτησης draw με την οποία απεικονίζεται στον καμβά το περιεχόμενο της μεταβλητής message στις συντεταγμένες της μεταβλητής position

```
def draw(canvas):
     canvas.draw_text(message, position , 40, “Red”)
```

Δημιουργία του frame «Test» πλάτους και ύψους (500x500)

```
frame = simplegui.create_frame(“Test”, width, height)
```

Δήλωση draw_handler το οποίο καλεί την συνάρτηση draw η οποία απεικονίζει το περιεχόμενο της μεταβλητής message στον καμβά

```
frame.set_draw_handler(draw)
```

Δημιουργία timer handler.Το χρονόμετρο καλεί την συνάρτηση tick κάθε 2000 milliseconds δηλαδή μεταβάλλει την θέση του message κάθε 2000 milliseconds

```
timer = simplegui.create_timer(time_in_milliseconds, tick)
```

Εκκίνηση frame

```
frame.start()
```

Εκκίνηση χρονομέτρου

```
timer.start()
```

4.3 ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΑΣΚΗΣΗ

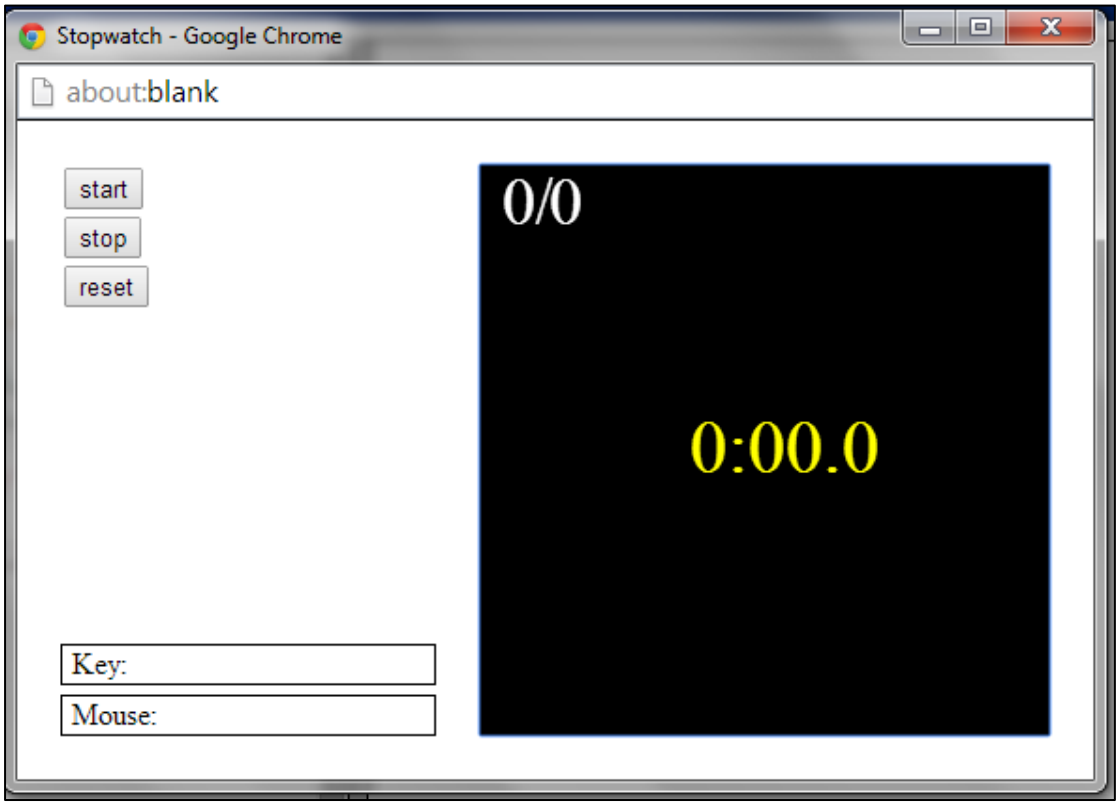
Στην τρίτη εβδομάδα του μαθήματος υλοποιήθηκε το STOPWATCH.Για το συγκεκριμένο παιχνίδι πρέπει να σχεδιαστεί στην περιοχή του καμβά ένα χρονόμετρο το οποίο απεικονίζει λεπτά,δευτερόλεπτα και δέκατα του δευτερολέπτου και στην περιοχή control area τρία buttons 'Start','Stop' ,'Reset' όπου θα ξεκινούν, θα σταματούν και θα μηδενίζουν το χρονόμετρο αντίστοιχα. Ο παίκτης θα πρέπει να θα σταματήσει το χρονόμετρο σε ακέραιο αριθμό για να κερδίσει .Επιπλέον στην περιοχή του καμβά θα πρέπει να εμφανίζεται πόσες φορές έχει πατήσει ο παίκτης το Stop button αλλά και πόσες φορές έχει κερδίσει. Το χρονόμετρο θα πρέπει να έχει την μορφή A:BC:D με τα A , C , D να είναι ψηφία εύρους 0-9 ενώ το B εύρος 0-5.

4.3.1 ΥΛΟΠΟΙΗΣΗ ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗΣ ΑΣΚΗΣΗΣ

```
Δήλωση simplegui
import simplegui

Αρχικοποίηση global μεταβλητών
time = 0 # αρχικοποίηση του χρόνου για το χρονόμετρο
win=0    #  αρχικοποίηση των νικών του παίκτη
tries=0  #αρχικοποίηση των προσπαθειών  του παίκτη
stop_control=True # Η μεταβλητή stop_control ελέγχει την έναρξη του χρονομέτρου
Όταν η τιμή του είναι True τότε το χρονόμετρο είναι σταματημένο

Βοηθητική συνάρτηση format που ορίζει την μορφή του χρονομέτρου σε λεπτά,
δευτερόλεπτα και δέκατα του δευτερολέπτου.
Η συνάρτηση επιστρέφει το αποτέλεσμα της σε μορφή string για να απεικονιστεί στον
καμβά
def format(t):
    c = t // 10
    d = t % 10
    b= c // 10
    c = c % 10
    a = b // 6
    b = b % 6
    return str(a) + ":" + str(b) + str(c) + ":" + str(d)

Ορισμός  συνάρτησης timer_handler η οποία αυξάνει το χρονόμετρο κατά ένα
def timer_handler():
    global time

    time=time+1

Ορισμός  συναρτήσεων buttons
Η συνάρτηση Start ξεκινάει το χρονόμετρο
def Start():
    global count_wins,stop_control

    stop_control = False
```

```
    timer.start()
Η συνάρτηση Stop σταματάει το χρονόμετρο
def Stop():
    global time,win,tries,stop_control
    timer.stop()

 Ενώ το χρονόμετρο τρέχει και ο παίκτης σταματήσει το χρονόμετρο σε ακέραιο αριθμό η μεταβλητή wins αυξάνει κατά ένα
  if stop_control == False
        tries +=1
        if time %10==0:
            win += 1
        stop_control = True # Το χρονόμετρο σταματάει

 Η συνάρτηση Reset   μηδενίζει τις νίκες, τις προσπάθειες και το χρονόμετρο και σταματάει το χρονόμετρο
def Reset():
    global time,win,tries
    win=0
    tries=0
    time=0
    timer.stop() # Η timer.stop καλείται γιατί ενώ θα μηδενιστεί το χρονόμετρο θα συνεχίσει να τρέχει

Ορισμός συνάρτησης draw handler η οποία απεικονίζει στον καμβά το χρονόμετρο ,τις προσπάθειες και τις νίκες του παίκτη
def draw_handler(canvas):

    canvas.draw_text(str(format(time)), ( 110, 162), 40, 'Yellow')
    canvas.draw_text(str(win) + '/' , (10, 30), 35, 'white')
    canvas.draw_text(str(tries), (35, 30), 35, 'white')

Δημιουργία frame Stopwatch
frame = simplegui.create_frame('Stopwatch', 300, 300)

Δημιουργία draw_handler το οποίο καλεί την συνάρτηση draw_handler
frame.set_draw_handler(draw_handler)

Δημιουργία χρονομέτρου το οποίο καλεί την συνάρτηση timer_handler κάθε 100 milliseconds για την απεικόνιση του χρονομέτρου στην οθόνη
timer = simplegui.create_timer(100, timer_handler)

Δημιουργία buttons για την εκκίνηση, τερματισμό και μηδενισμό του χρονομέτρου
frame.add_button('start', Start)
frame.add_button('stop', Stop)
frame.add_button('reset', Reset)

Εκκίνηση frame
frame.start()
```

ΕΞΟΔΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ



Στην παρακάτω εικόνα στο πάνω μέρος του καμβά απεικονίζονται στα αριστερά 'οι νίκες 'και στα δεξιά 'οι προσπάθειες '.

5.PONG

Στο κεφάλαιο αυτό παρουσιάζουμε την υλοποίηση της προγραμματιστικής άσκησης Pong ή όπως είναι γνωστό πινγκ-πονγκ.Στο συγκεκριμένο κεφάλαιο παρουσιάζονται οι εντολές του Codeskulptor για την κίνηση αντικειμένων στον καμβά όπως και για τον έλεγχο αντικειμένων από το πληκτρολόγιο.

5.1 ΛΙΣΤΕΣ

Λίστα είναι μία μεταβλητή ακολουθία οποιουδήποτε τύπου δεδομένων.

1.Δημιουργία κενής λίστας

```
list =[]
```

2.Δημιουργία λίστας με βήμα

```
range( stop)
range(start , stop)
range(start , stop, step)
```

Παραδείγματα:

```
print range(5)
[0, 1, 2, 3, 4]
```

Εκτυπώνεται μία λίστα από 0 έως το 4

```
print range(1, 10)
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Εκτυπώνεται μία λίστα από το 0 έως το 9

```
print range(10, 100, 10)
[10, 20 ,30 , 40, 50, 60, 70, 80 , 90]
```

Εκτυπώνεται μία λίστα απο το 10 έως το 99 με βήμα 10

3.Προσθήκη στοιχείων σε λίστα

List.append(x)

Παραδείγματα:

```
x=[1, 2, 3]
x.append(4)
print x
[1, 2, 3, 4]
```

Προστίθεται στην λίστα x ο αριθμός 4

```
x.append([5 ,6, 7])
print x
[1, 2, 3, 4 , [5, 6, 7]]
```

Προστίθεται στην λίστα x οι αριθμοί 5, 6 ,7

4.Εισαγωγή στοιχείων στην λίστα

List.insert(i, x)

Παραδείγματα:

```
x = [“a”, “b”, “c”]
x.insert(1, “z”)
print x
[‘a’, 'z', 'b', 'c']
```

Εισάγεται το στοιχείο z στην λίστα x

5.Μετακίνηση στοιχείων απο την λίστα

List.remove(x)

Παραδείγματα:

```
x = [“a”, “b”, “c”, “b”]
x.remove(“b”)
print x
```

['a', 'c', 'b']

Μετακινείται το πρώτο b από την λίστα

5.2 Έλεγχος αντικειμένων απο το πληκτρολόγιο

ΣΥΝΤΑΞΗ:

frame.set_keydown_handler (keydown_handler)

frame.set_keyup_handler (keyup_handler)

Με τις εντολές frame.set_keydown_handler και frame.set_keyup_handler ο χρήστης ελέγχει αντικείμενα στην περιοχή του καμβά με την βοήθεια του πληκτρολογίου. Όταν ο χρήστης χρησιμοποιεί κάποιο πλήκτρο καλείται για μία φορά η keydown_handler ενώ όταν ο χρήστης αφήσει το συγκεκριμένο πλήκτρο καλείται η keyup_handler. Η παράμετρος που θα δέχεται η συνάρτηση keydown_handler και keyup_handler θα πρέπει να είναι κάποιος χαρακτήρας του πληκτρολογίου με την μορφή ακέραιου αριθμού. Οι χαρακτήρες του πληκτρολογίου που αναγνωρίζονται από το Codeskulptor είναι «space», «right» ,«left» «up » ,«down», «a-z» «A-Z» ,«0-9».Με την εντολή simplegui.KEY_MAP το Codeskulptor αντιστοιχεί κάποιον από τους χαρακτήρες του πληκτρολογίου σε κάποιον ακέραιο αριθμό.

Παράδειγμα:

```
import simplegui

print simplegui.KEY_MAP ['up']
```

Έξοδος: 38

Παρακάτω δίνεται ένα πρόγραμμα ελέγχου αντικειμένου.Συγκεκριμένα χρησιμοποιώντας την εντολή canvas.draw_circle σχεδιάζεται μία μπάλα στο κέντρο του καμβά.Ο χρήστης έχει την δυνατότητα με την χρήση των χαρακτήρων right,up,down και left του πληκτρολογίου(βελάκια) να μετακινήσει την μπάλα αριστερά,δεξιά,πάνω και κάτω.

```
Δήλωση simplegui
import simplegui

Δήλωση μήκους και ύψους του καμβά σε pixels
WIDTH = 600
HEIGHT = 400
```

```
Δήλωση ακτίνας της μπάλας
BALL_RADIUS =20

Δήλωση της θέση της μπάλας στο μέσο του καμβά στην εκκίνηση του προγράμματος
Η θέση της μπάλας δηλώνεται σαν ένα σημείο (x, y)σε ένα σύστημα συντεταγμένων
ball_pos = [WIDTH/2 , HEIGHT/2]

Ορισμός της συνάρτησης draw handler που σχεδιάζει μία μπάλα
def draw(canvas):
    canvas.draw_circle(ball_pos, BALL_RADIUS, 2, "Red", "White")


Ορισμός της συνάρτησης  για το keyboard handler
Η συνάρτηση keydown παίρνει σαν παράμετρο το key στο οποίο εκχωρείται ένας
ακέραιος με την εντολή simplegui.KEY_MAP.
Η μεταβλητή vel αντιπροσωπεύει pixels
Με την συνθήκη if ελέγχεται πιο πλήκτρο χρησιμοποιεί ο χρήστης. Όταν
χρησιμοποιείται το πλήκτρο left τότε μειώνεται το ball_pos[0] δηλαδή την οριζόντια
θέση της μπάλας κατά τέσσερα pixels,αντίθετα όταν χρησιμοποιείται το πλήκτρο right
αυξάνεται η οριζόντια θέση της μπάλας κατά τέσσερα pixels.Το πλήκτρο down αυξάνει
την κάθετη θέση της μπάλας ball_pos[1] κατά τέσσερα pixels ενώ το πλήκτρο up
μειώνει την κάθετη θέση της μπάλας κατά τέσσερα pixels.

def keydown(key):
    vel = 4
    if key == simplegui.KEY_MAP["left"]:
      ball_pos[0] -= vel
    elif key == simplegui.KEY_MAP["right"]:
       ball_pos[0] += vel
    elif key == simplegui.KEY_MAP["down"]:
        ball_pos[1] += vel
    elif key == simplegui.KEY_MAP["up"]:
      ball_pos[1] -=vel



Δημιουργία frame
frame = simplegui.create_frame("ball control", WIDTH, HEIGHT)

Δήλωση draw handler το οποίο καλεί την συνάρτηση draw
frame.set_draw_handler(draw)

Δήλωση keyboard handler το οποίο καλεί την συνάρτηση keydown
frame.set_keydown_handler(keydown)

Εκκίνηση frame
frame.start()
```

5.3 Η συνάρτηση keydown

Η συνάρτηση keydown παίρνει σαν όρισμα την παράμετρο key.Η key δεν είναι κάποιο γράμμα του πληκτρολογίου απλά είναι μία παράμετρος όπου αντιστοιχείται στο εκάστοτε πλήκτρο του πληκτρολογίου που χρησιμοποιείται.

```
def keydown(key):
    vel = 4
 if key == simplegui.KEY_MAP[“left”]

   ball_pos[0] - = vel
elif key == simplegui.KEY_MAP[“right”]:
        ball_pos[0] += vel
```

Η εντολή simplegui.KEY_MAP βρίσκει το αντίστοιχο key του πληκτρολογίου που έχει χρησιμοποιήθει.Στην παραπάνω περίπτωση το key είναι “left” το αριστερό βελάκι του πληκτρολογίου.Αντίστοιχα το “right” είναι το δεξί βελάκι του πληκτρολογίου.

Ο καμβάς είναι ένα σύστημα συντεταγμένων όπου ο οριζόντιος άξονας είναι το x και ο κάθετος άξονας είναι το y.Στο παραπάνω πρόγραμμα ορίσαμε την θέση της μπάλας σαν μία λίστα δύο στοιχείων ball_pos[x, y] που ισούνται με (300,200) pixels αντίστοιχα.Οταν ο χρήστης χρησιμοποιεί το αριστερό βελάκι του πληκτρολογίου τότε με την εντολή ball_pos [0] - = vel το πρόγραμμα μετακινεί το στοιχείο που βρίσκεται στην θέση μηδέν της λίστας ball_pos δηλαδή το x προς τα αριστερά κατά 4 pixels. Αντίστοιχα το δεξί βελάκι μετακινεί την θέση του x προς τα δεξιά κατά 4 pixels.

```
elif key == simplegui.KEY_MAP[“down”]:
    ball_pos[1] += vel
 elif key == simplegui.KEY_MAP[“up”]:
    ball_pos[1] -=vel
```

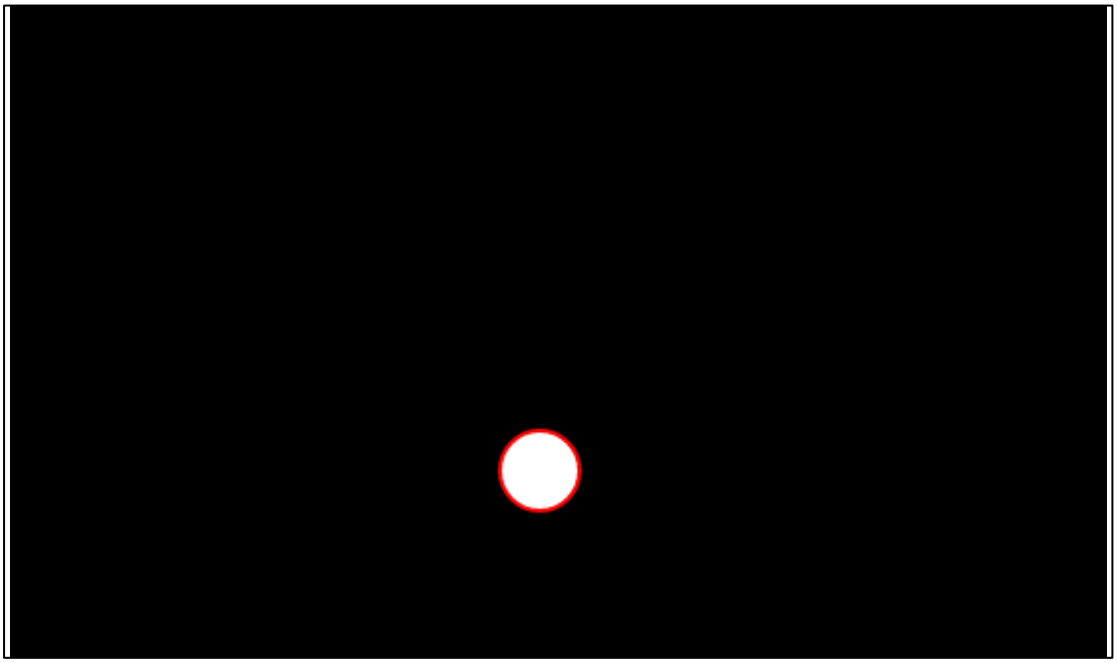
Ίδια ακριβώς λογική και για τα βελάκια “up” και “down” με την μόνη διαφορά ότι μετακινείται το πρώτο στοιχείο της λίστας ball_pos δηλαδή μεταβάλλονται οι συντεταγμένες του άξονα y κατά 4 pixels.

5.4 Κίνηση αντικειμένων στον καμβά

Το παρακάτω πρόγραμμα εμφανίζει μία μπάλα στον καμβά και ελέγχει την κατεύθυνση της κίνησης της.

```
Δήλωση simplegui
import simplegui

Δήλωση μήκους και ύψους του καμβά σε pixels
WIDTH = 600
HEIGHT = 400

Δήλωση ακτίνας μπάλας
BALL_RADIUS = 20

Δήλωση αρχικής θέσης της μπάλας στο μέσο του καμβά
Η θέση της μπάλας δηλώνεται σαν μία λίστα δύο στοιχείων
ball_pos = [WIDTH/2, HEIGHT/2]
Δήλωση  της λίστας vel η οποία αναπαριστά την  ταχύτητας της μπάλας σε pixels
vel = [0, 1]

Ορισμός συνάρτησης draw handler η οποία απεικονίζει την μπάλα στην οθόνη.
Η  εντολή  ball_pos[0]  += vel[0] ανανεώνει την θέση της μπάλας στον οριζόντιο
άξονα του καμβά και η εντολή  ball_pos[1]  += vel[1 ] ανανεώνει την θέση της
μπάλας στον κάθετο άξονα του καμβά.
Το vel [0] είναι το πρώτο στοιχείο της λίστας vel το οποίο είναι μηδέν άρα η θέση της
μπάλας στον οριζόντιο άξονα μένει αμετάβλητη ενώ το vel [1] είναι το δεύτερο στοιχείο
της λίστας vel το οποίο είναι ένα άρα η θέση της μπάλας στον κάθετο άξονα αυξάνεται
κατά ένα pixel.
def draw(canvas):
    ball_pos[0]  += vel[0]
    ball_pos[1]  += vel[1]

   canvas.draw_circle(ball_pos, BALL_RADIUS, 2, "Red","White")

Δημιουργία frame
frame = simplegui.create_frame("Motion",WIDTH, HEIGHT)

Δημιουργία draw_handler το οποίο καλεί την συνάρτηση draw
frame.set_draw_handler(draw)

Εκκίνηση frame
frame.start()
```

ΕΞΟΔΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ



Η έξοδος του προγράμματος εμφανίζει μια μπάλα στο μέσο του καμβά η οποία κατευθύνεται στο κάτω μέρος του καμβά. Αυτό είναι απολύτως λογικό αφού ανανεώθηκε η θέση της μπάλας κατά ένα pixel μόνο στον κάθετο άξονα

Κατεύθυνση της μπάλας στο πάνω μέρος του καμβά

Για να κατευθυνθεί η μπάλα στο πάνω μέρος του καμβά πρέπει να μειωθεί η θέση της μπάλας στον κάθετο άξονα κατά ένα pixel ενώ η ταχύτητα παραμένει ίδια.

vel = [0, 1]

ball_pos [0] + = vel [0]

ball_pos [1] - = vel [1]

Κατεύθυνση της μπάλας στο αριστερό μέρος του καμβά

Για να κατευθυνθεί η μπάλα στο αριστερό μέρος του καμβά πρέπει να μειωθεί η θέση της μπάλας στον οριζόντιο άξονα κατά ένα pixel ενώ η ταχύτητα του vel [0] γίνεται ένα και του vel [1] μηδέν.

vel = [1, 0]

ball_pos [0] -= vel [0]

ball_pos [1] += vel [1]

Κατεύθυνση της μπάλας στο δεξί μέρος του καμβά

Για να κατευθυνθεί η μπάλα στο δεξί μέρος του καμβά πρέπει να αυξηθεί η θέση της μπάλας στον οριζόντιο άξονα κατά ένα pixel ενώ η ταχύτητα του vel [0] γίνεται ένα και του vel [1] μηδέν.

vel = [1, 0]

ball_pos [0] -= vel [0]

ball_pos [1] += vel [1]

Διαγώνια κατεύθυνση της μπάλας στον καμβά

Για την κατεύθυνση της μπάλας διαγωνίως πρέπει να ανανεωθεί η θέση της μπάλας σε συνάρτηση με την ταχύτητα και στον οριζόντιο αλλά και στον κάθετο άξονα. Αυτό επιτυγχάνεται δίνοντας τιμή ένα στο vel[0] και vel[1].

vel = [1, 1]

ball_pos [0] += vel [0]

ball_pos [1] += vel [1]

ΕΞΟΔΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

Όπως παρατηρούμε από την έξοδο του προγράμματος η μπάλα κατευθύνεται διαγωνίως στην κάτω δεξιά πλευρά του καμβά. Για να αλλάξουμε την κατεύθυνση της μπάλας στην πάνω δεξιά ή κάτω αριστερά ή πάνω αριστερά πλευρά του καμβά αρκεί να μειώσουμε ή να αυξήσουμε την θέση της μπάλας σε συνάρτηση με την ταχύτητα.

Διαγώνια κατεύθυνση στην πάνω αριστερή πλευρά του καμβά

vel = [1, 1]

ball_pos [0] -= vel [0]

ball_pos [1] -= vel [1]

Διαγώνια κατεύθυνση στην πάνω δεξιά πλευρά του καμβά

vel = [1, 1]

ball_pos [0] += vel [0]

ball_pos [1] -= vel [1]

Διαγώνια κατεύθυνση στην κάτω αριστερή πλευρά του καμβά

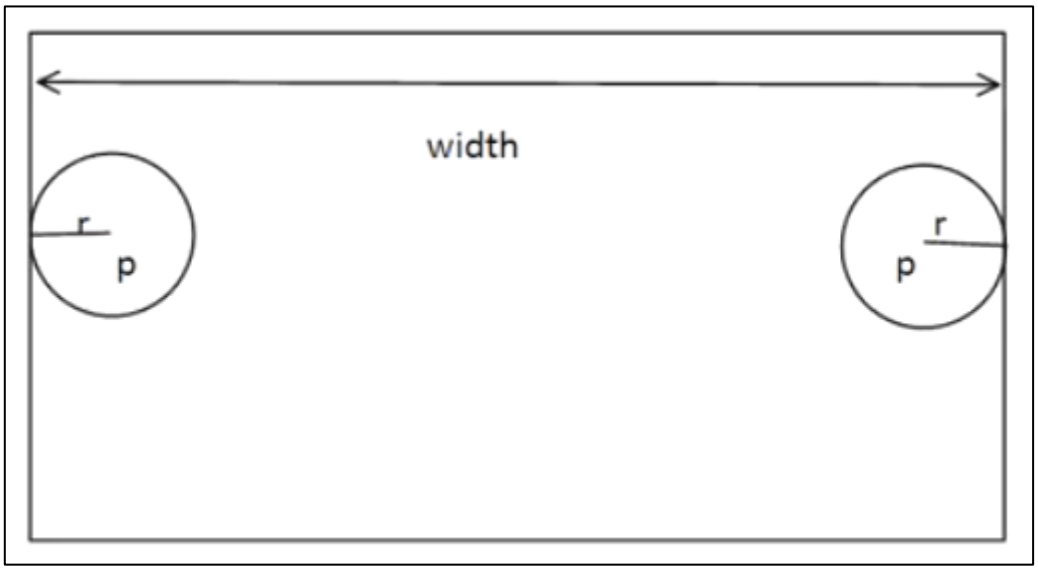
vel = [1, 1]

ball_pos [0] -= vel [0]

ball_pos [1] += vel [1]

5.5 Συγκρούσεις και αντανακλάσεις αντικειμένων στον καμβά

Στο προηγούμενο παράδειγμα δείξαμε πώς μπορεί να ελέγχει η κατεύθυνση της μπάλας σε συνάρτηση της ταχύτητας της. Το πρόβλημα που προκύπτει όμως είναι ότι η μπάλα δεν αντανακλάται από τον καμβά άρα χάνεται από την οθόνη. Παρακάτω δίνονται οι τρόποι επίλυσης του προβλήματος ανάλογα με την κατεύθυνση της μπάλας στον καμβά.



Σύγκρουση της μπάλας με κέντρο p και ακτίνα(radius) r με τον αριστερό τοίχο του καμβά

p[0]<=r

Σύγκρουση της μπάλας με κέντρο p και ακτίνα(radius) r με τον δεξί τοίχο του καμβά

p[0]>=(width-1)-r

Σύγκρουση σημείου p με τον αριστερό τοίχο του καμβά

p[0]<=0

Σύγκρουση σημείου p με τον δεξί τοίχο του καμβά

p[0]>=(width-1)

Ταχύτητα μπάλας όταν αντανακλάται απο τον αριστερό τοίχο.

v[0] = -v[0]

v[1] = v[1]

Ανανέωση κίνησης αντικειμένου

p[0]= p[0]+ v[0]

p[1]= p[1] +v[1]

5.5.1 Αντανάκλαση μπάλας στην αριστερή πλευρά του καμβά

```
Δήλωση simplegui
import simplegui

Δήλωση global μεταβλητών για το πλάτος και το ύψος του καμβά
WIDTH = 600
HEIGHT = 400

Δήλωση ακτίνας μπάλας
BALL_RADIUS = 20

Δήλωση αρχικής θέσης της μπάλας στον καμβά
ball_pos =[WIDTH / 2, HEIGHT / 2]

Δήλωση ταχύτητας της μπάλας
vel = [-40.0/60.0 , 5.0/60.0]

Δήλωση συνάρτησης event handler
def draw(canvas):

Ανανέωση της θέσης της μπάλας σε συνάρτηση με την ταχύτητα(vel)
    ball_pos[0] += vel[0]
    ball_pos[1] += vel[1]

Πρόσκρουση και αντανάκλαση της μπάλας στον αριστερό τοίχο
Η ball_pos [0] είναι η θέση που έχει η μπάλα στον οριζόντιο άξονα του καμβά.
Όταν η θέση που έχει η μπάλα γίνει μικρότερη ή ίση με την ακτίνα της μπάλας τότε η
μπάλα αντανακλάται από τον αριστερό τοίχο.
   if ball_pos[0] <=BALL_RADIUS:
   vel[0] = - vel[0]

Σχεδιασμός μπάλας
canvas.draw_circle(ball_pos, BALL_RADIUS, 2, "Red", "White")

Δημιουργία frame
frame = simplegui.create_frame("BALL", WIDTH, HEIGHT)
```

```
Δήλωση event handler το οποίο καλεί την συνάρτηση draw
frame.set_draw_handler(draw)

Εκκίνηση frame
frame.start()
```

ΕΞΟΔΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

5.6 ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΑΣΚΗΣΗ

Την τέταρτη εβδομάδα του μαθήματος υλοποιήθηκε το Pong.Το Pong είναι ένα δυσδιάστατο παιχνίδι που προσομοιώνει το πινγκ-πονγκ.Στο συγκεκριμένο παιχνίδι συμμετέχουν δύο παίκτες οι οποίοι με ένα κουπί(paddle) τα οποία είναι σχεδιασμένα στην δεξιά και αριστερή πλευρά του καμβά προσπαθούν να χτυπήσουν την μπάλα χωρίς να γίνει έρθει σε επαφή η μπάλα με τον δεξί ή αριστερό τοίχο του καμβά.

Πιο αναλυτικά:

Πρέπει να σχεδιαστούν δύο κουπιά(paddle1 και paddle2),ένα για κάθε παίκτη και μία μπάλα στο κέντρο του καμβά.Ο κάθε παίκτης πρέπει να χτυπήσει την μπάλα με ένα paddle το οποίο κινείται κατακόρυφα του καμβά.Εάν κάποιος παίκτης αποτύχει να χτυπήσει την μπάλα τότε δίνεται ένας πόντος στον αντίπαλο.Τα paddle1 και paddle2 πρέπει να ελέγχονται από το πληκτρολόγιο με ένα keyboard handler.Η ταχύτητα της μπάλας πρέπει αυξομειώνεται όταν η μπάλα χτυπάει σε ένα paddle.

5.6.1 ΥΛΟΠΟΙΗΣΗ ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗΣ ΑΣΚΗΣΗΣ

```
Δήλωση simplegui
import simplegui
import random

Δήλωση global μεταβλητών
WIDTH = 600  #Μήκος του καμβά
HEIGHT = 400   #Υψος  του καμβά
BALL_RADIUS = 20 #Ακτίνα της μπάλας
PAD_WIDTH = 8  #Μήκος του paddle
PAD_HEIGHT = 80 # Ύψος του paddle
HALF_PAD_WIDTH = PAD_WIDTH / 2
HALF_PAD_HEIGHT = PAD_HEIGHT / 2
left = False
right = True
paddle1_pos = HEIGHT/2  #Θέση του paddle1
paddle2_pos = HEIGHT/2  # Θέση του paddle2

Η συνάρτηση spawn_ball παίρνει σαν παράμετρο την κατεύθυνση της μπάλας
Διασκορπίζει την μπάλα είτε στην αριστερή είτε στην δεξιά πλευρά του καμβά δίνοντας
στην ταχύτητα της μία τυχαία τιμή με την συνάρτηση  random.randrange.
Η θέση της μπάλας(ball_pos) και η ταχύτητα(ball_vel) της ορίζονται σαν global για να
χρησιμοποιηθούν οπουδήποτε στο πρόγραμμα.
def spawn_ball(d):
    global ball_pos, ball_vel
    ball_pos=[WIDTH/2,HEIGHT/2]

    ball_vel[1] = -random.randrange(60, 180)/60
    if  d==right:
        ball_vel[0]=random.randrange(120, 240)/60
        ball_pos[0] +=ball_vel[0]
    elif d==left:
```

```
        ball_vel[0]=-random.randrange(120, 240)/60
        ball_pos[0] +=ball_vel[0]

Η συνάρτηση new_game εκκινεί το παιχνίδι και αρχικοποιεί τις μεταβλητές του προγράμματος
Η θέση της μπάλας στην εκκίνηση του παιχνιδιού ορίζεται στο κέντρο του καμβά

def new_game():
    global paddle1_pos, paddle2_pos, paddle1_vel, paddle2_vel,ball_vel,ball_pos
    global score1, score2

    paddle1_pos=HEIGHT/2
    paddle2_pos=HEIGHT/2
    paddle1_vel = 0  #η αρχική ταχύτητα του paddle1 είναι 0
    paddle2_vel = 0  #η αρχική ταχύτητα του paddle2 είναι 0
    score1=0  #το score των παικτών μηδενίζεται
    score2=0  #το score των παικτών μηδενίζεται
    ball_pos=[WIDTH/2,HEIGHT/2] #η  εκκίνηση της μπάλας γίνεται από το κέντρο του καμβά.
    ball_vel = [random.randrange(120, 180)/60,-random.randrange(120, 240)/60] #Η random.randrange δίνει τυχαίες τιμές στην ταχύτητα της μπάλας

Δημιουργία συνάρτησης draw με την οποία απεικονίζονται η μπάλα, τα paddle1 και paddle2,το score1 και score2
def draw(canvas):
    global score1, score2, paddle1_pos, paddle2_pos, ball_pos,
ball_vel,paddle1_vel,paddle2_vel


    # σχεδιασμός γραμμών που σχηματίζουν το pong
    canvas.draw_line([WIDTH / 2, 0],[WIDTH / 2, HEIGHT], 1, "White")
    canvas.draw_line([PAD_WIDTH, 0],[PAD_WIDTH, HEIGHT], 1, "White")
    canvas.draw_line([WIDTH - PAD_WIDTH, 0],[WIDTH - PAD_WIDTH,
HEIGHT], 1, "White")

    # ανανέωση θέσης της μπάλας
    ball_pos[0] +=ball_vel[0]
    ball_pos[1] +=ball_vel[1]

    # σχεδιασμός της μπάλας
    canvas.draw_circle(ball_pos, BALL_RADIUS, 2, "yellow", "White")

    #σχεδιασμός των paddle1  και paddle2
    canvas.draw_line([3, paddle1_pos+50],[2, paddle1_pos-60], 15, "White")
    canvas.draw_line([597, paddle2_pos+50],[597, paddle2_pos-60], 15, "White")

    #ανανέωση της κατακόρυφης θέσης των  paddle1 και paddle2 για την παραμονή
    τους στα όρια του καμβά όταν ο χρήστης μετακινεί τα paddle1 και  paddle2 με την
   βοήθεια του πληκτρολογίου
    if paddle1_pos<=60 and paddle1_vel < 0: # όταν το paddle1 βρίσκεται στο αριστερό
```

```
πάνω άκρο του καμβά η ταχύτητα του μηδενίζεται.
    paddle1_vel=0
elif paddle1_pos>=350 and paddle1_vel > 0 : # όταν το paddle1 βρίσκεται στο
αριστερό κάτω άκρο του καμβά η ταχύτητα του μηδενίζεται.
    paddle1_vel=0
elif paddle2_pos<=60 and paddle2_vel < 0:  # όταν το paddle2  βρίσκεται στο δεξί
πάνω άκρο του καμβά η ταχύτητα του μηδενίζεται.
    paddle2_vel=0
elif paddle2_pos>=350 and paddle2_vel > 0 : # όταν το paddle2  βρίσκεται στο δεξί
κάτω άκρο του καμβά η ταχύτητα του μηδενίζεται.
    paddle2_vel=0

#ανανέωση της θέσης του paddle1 και  paddle2
paddle1_pos = paddle1_pos + paddle1_vel
paddle2_pos = paddle2_pos + paddle2_vel



#ανανέωση της ταχύτητας της μπάλας εάν κατευθύνεται στο πάνω μέρος του
καμβά για την παραμονή της μπάλας στα όρια του καμβά όταν αυτή προσκρούει στο
πάνω μέρος του καμβά.
if ball_pos[1]<=BALL_RADIUS:
    ball_vel[1]=-ball_vel[1]



#ανανέωση της ταχύτητας της μπάλας εάν κατευθύνεται στο κάτω μέρος του
καμβά για την παραμονή της μπάλας στα όρια του καμβά όταν αυτή προσκρούει στο
κάνω μέρος του καμβά.

elif ball_pos[1]>=(HEIGHT-1)-BALL_RADIUS:
    ball_vel[1]=-ball_vel[1]

#σύγκρουση της μπάλας με τον αριστερό τοίχο
if ball_pos[0]<=BALL_RADIUS:

   if ball_pos[1] < (paddle1_pos - HALF_PAD_HEIGHT) or
      ball_pos[1]>(paddle1_pos + HALF_PAD_HEIGHT):
      score2+=1
     ball_pos=[WIDTH/2,HEIGHT/2] #αρχική θέση της μπάλας
     spawn_ball(right)  # καλείται η συνάρτηση spawn_ball που κατευθύνει την
     μπάλα στην δεξιά πλευρά του καμβά

   #όταν η μπάλα έρθει σε επαφή με το paddle1 ανανεώνεται η ταχύτητα της για να
   αλλάζει κατεύθυνση
    else:
        ball_vel[0] =(random.randrange(120, 240)/60)*2
```

```
    #σύγκρουση της μπάλας με τον δεξί τοίχο
    if ball_pos[0]>=(WIDTH-1)-BALL_RADIUS:
       if ball_pos[1] < (paddle2_pos - HALF_PAD_HEIGHT) or ball_pos[1] >
(paddle2_pos

+ HALF_PAD_HEIGHT):
          score1+=1
          ball_pos=[WIDTH/2,HEIGHT/2] #αρχική θέση της μπάλας
          spawn_ball(left) # καλείται η συνάρτηση spawn_ball που κατευθύνει την
          μπάλα στην αριστερή πλευρά του καμβά

       #όταν η μπάλα έρθει σε επαφή με το paddle2 ανανεώνεται η ταχύτητα της και
       αλλάζει κατεύθυνση
        else:
            ball_vel[0] = (-random.randrange(170, 240)/60)*1

    # σχεδιασμός του score1  και score2 στον καμβά
    canvas.draw_text(str(score1), (50, 45), 40, 'white')
    canvas.draw_text(str(score2), (520, 45), 40, 'white')

 Ορισμός της συνάρτησης  keydown για τον έλεγχο του paddle1  και paddle2 από το
πληκτρολόγιο
Η ταχύτητα  paddle1_vel και  paddle2_vel μπορεί να αλλάξει ανάλογα με το πόσο
γρήγορα θέλουμε να κινούνται τα paddle1 και paddle2
def keydown(key):
    global paddle1_vel, paddle2_vel

    if key == simplegui.KEY_MAP['w']:
        paddle1_vel = -5
    elif key == simplegui.KEY_MAP['s']:
        paddle1_vel = 5
    elif key == simplegui.KEY_MAP['up']:
        paddle2_vel = -5


 elif key == simplegui.KEY_MAP['down']:
        paddle2_vel = 5

Ορισμός της συνάρτησης  keyup η οποία μηδενίζει την ταχύτητα των paddle1  και
paddle2
Η συνάρτηση keyup καλείται την στιγμή που ο χρήστης αφήνει κάποιο πλήκτρο από τα
παρακάτω που ορίσθηκαν για τον έλεγχο των paddle1 και paddle.Η ταχύτητα των
paddle1 και paddle2 μηδενίζεται.

def keyup(key):
    global paddle1_vel, paddle2_vel
    if key == simplegui.KEY_MAP['w']:
        paddle1_vel = 0
    elif key == simplegui.KEY_MAP['s']:
        paddle1_vel = 0
```

```
    elif key == simplegui.KEY_MAP['up']:
        paddle2_vel = 0
    elif key == simplegui.KEY_MAP['down']:
        paddle2_vel = 0


# Δημιουργία frame
frame = simplegui.create_frame("Pong", WIDTH, HEIGHT)

# Δήλωση draw_handler το οποίο καλεί την συνάρτηση draw
frame.set_draw_handler(draw)

# Δήλωση keydown_ handler το οποίο καλεί την συνάρτηση keydown
frame.set_keydown_handler(keydown)

# Δήλωση keyup handler το οποίο καλεί την συνάρτηση keyup
frame.set_keyup_handler(keyup)

# Δημιουργία button Restart το οποίο εκκινεί το παιχνίδι
frame.add_button("Restart", new_game)

# Καλείται η συνάρτηση new_game
new_game()

# Εκκίνηση frame
frame.start()
```

ΕΞΟΔΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

6.MEMORY

Στο κεφάλαιο αυτό παρουσιάζουμε την υλοποίηση της προγραμματιστικής άσκησης Memory.Για τις ανάγκες υλοποίησης της προγραμματιστικής άσκησης παρουσιάζονται σε αυτό το κεφάλαιο οι εντολές του Codeskulptor για έλεγχο αντικειμένων από το ποντίκι όπως και φόρτωση εικόνων από το ίντερνετ και απεικόνιση τους στον καμβά.

6.1 Έλεγχος αντικειμένων με το ποντίκι

ΣΥΝΤΑΞΗ:

```
frame.set_mouseclick_handler(mouseclick_handler)
```

Η παραπάνω εντολή προσθέτει στο πρόγραμμα ένα event handler για έλεγχο αντικειμένων με το ποντίκι.Η συνάρτηση του event handler ορίζεται ως εξής:

```
def mouseclick_handler(position):
    …
```

με παράμετρο συνάρτησης το position το οποίο είναι ένα ζευγάρι συντεταγμένων στον καμβά

Στο παρακάτω πρόγραμμα ορίσαμε ένα event handler όπου ο χρήστης μπορεί με ένα "κλικ" του ποντικιού να αλλάξει την θέση ενός αντικειμένου στον καμβά.

```
Δήλωση simplegui
import simplegui

Δήλωση βιβλιοθήκης math για τον υπολογισμό μαθηματικών συναρτήσεων
import math

Δήλωση μήκους και ύψους του καμβά
WIDTH = 450
HEIGHT = 300

Δήλωση αρχικής θέσης της μπάλας στον καμβά
ball_pos = [WIDTH / 2, HEIGHT / 2]

Δήλωση ακτίνας μπάλας
BALL_RADIUS = 15

Δήλωση χρώματος μπάλας
ball_color = "Red"


Ορισμός συνάρτησης mouseclick η οποία παίρνει σαν παράμετρο ένα ζευγάρι
συντεταγμένων που στην συγκεκριμένη περίπτωση είναι μία λίστα συντεταγμένων που
```

```
έχει η μπάλα όταν ο χρήστης αλλάζει την θέση της στον καμβά
def mouseclick(pos):
global ball_pos
    ball_pos = list(pos)



Συνάρτηση draw για τον σχεδιασμό της μπάλας
def draw(canvas):
  canvas.draw_cicle(ball_pos, BALL_RADIUS, 1, "black", ball_color)


Δημιουργία frame και ορισμός του χρώματος του background του καμβά σε λευκό
frame = simplegui.create_frame("Mouse selection", WIDTH, HEIGHT)
frame.set_canvas_background("White")

Δήλωση event handler τα οποία καλούν τις συναρτήσεις mouseclick και draw
frame.set_mouseclick_handler(mouseclick)
frame.set_draw_handler(draw)

Εκκίνιση frame
frame.start()
```

Κάθε φορά που ο χρήστης κάνει "κλικ" με το ποντίκι σε κάποιο σημείο του καμβά τότε το πρόγραμμα καλεί την συνάρτηση mouseclick η οποία παίρνει σαν παράμετρο τις συντεταγμένες αυτού του σημείου και ανανεώνει κάθε φορά τη θέση της μπάλας στον καμβά.

Ας δούμε τώρα πως μπορούμε να αλλάξουμε το χρώμα της μπάλας με event_handler. Στο πρόγραμμα μας θέλουμε κάθε φορά που ο χρήστης κάνει "κλικ" στο εσωτερικό της μπάλας το χρώμα της να αλλάζει απο κόκκινο σε πράσινο.Για να γίνει κάτι τέτοιο πρέπει να χρησιμοποιήσουμε τον τύπο του πυθαγόρειου θεωρήματος.

Ο τύπος του πυθαγόρειου θεωρήματος υπολογίζει την απόσταση της προηγούμενης θέσης της μπάλας με την καινούργια. Αν η απόσταση των δύο σημείων είναι μικρότερη ή ίσον της ακτίνας της μπάλας τότε η μπάλα γίνεται πράσινη αλλιώς παραμένει κόκκινη.Την προηγούμενη θέση της μπάλας μας την δίνει η ball_pos και την καινούργια θέση το pos.

```
Δήλωση global μεταβλητών
import simplegui

Δήλωση βιβλιοθήκης math για τον υπολογισμό της συνάρτησης του πυθαγορείου
θεωρήματος
import math
```

```
Δήλωση του πλάτους και του ύψους του καμβά
WIDTH = 450
HEIGHT = 300

Δήλωση της θέσης της μπάλας στο μέσο του καμβά
ball_pos = [WIDTH / 2, HEIGHT / 2]

Δήλωση της ακτίνας και του χρώματος της μπάλας
BALL_RADIUS = 15
ball_color = "Red"

Υπολογισμός συνάρτησης πυθαγορείου θεωρήματος
def distance(p, q):
    return math.sqrt( (p[0] – q[0] ) ** 2 + (p[1] – q[1]) ** 2)


Η συνάρτηση click την απόσταση της θέσης της μπάλας(pos) από την προηγούμενη
θέση που είχε(ball_pos)
Εάν η απόσταση μεταξύ είναι μικρότερη του 15 δηλαδή της ακτίνας της μπάλας τότε το
χρώμα της γίνεται πράσινο
def click(pos):
    global ball_pos, ball_color

    if   distance(pos, ball_pos)  < BALL_RADIUS:
        ball_color="Green"
    else:
       ball_pos = list(pos)

Η συνάρτηση draw σχεδιάζει την μπάλα στον καμβά
def draw(canvas):
  canvas.draw_cicle(ball_pos, BALL_RADIUS, 1, "black", ball_color)

Δημιουργία frame και ορισμός του background σε λευκό χρώμα
frame = simplegui.create_frame("Mouse selection", WIDTH, HEIGHT)
frame.set_canvas_background("White")

Δήλωση event handler τα οποία καλούν τις συναρτήσεις click και draw
frame.set_mouseclick_handler(click)
frame.set_draw_handler(draw)

Εκκίνηση frame
frame.start()
```

6.2 DICTIONARIES

Στην Python ένα λεξικό(dictionary) είναι ένας τύπος αντικειμένων σαν λίστα όμως αντί για στοιχεία αποτελείται απο ένα σύνολο μη ταξινομημένων κλειδιών(keys).Κάθε κλειδί είναι μοναδικό και αντιστοιχείται σε μία μεταβλητή.Ένα λεξικό ξεκινάει και τελειώνει με άγκιστρα.Τα ζεύγη κλειδιού μεταβλητής χωρίζονται με κόμματα ενώ το κλειδί χωρίζεται από την μεταβλητή με άνω κάτω τελεία.

1. dict= {}

Δημιουργία κενού λεξικό

2. a_dict = {1: 'a' , 2: 'b' , 8:'c'}

Λεξικό με τρία ζεύγη κλειδιών

3.a_dict[key]

a_dict.get(key)

Παράδειγμα:

```
a_dict={1: 'a', 2: 'b', 3: 'c'}
print  a_dict.get(2)
```

Εκτυπώνει την μεταβλητή που περιέχεται στο key

Έξοδος: 'b'

4. a_dict [key] = value

Παράδειγμα:

```
a_dict = {1: 'a', 2: 'b', 3: 'c'}
a_dict[2] = 'w'
print a_dict
```

Αντικατάσταση της μεταβλητής 'b' που περιέχεται στο κλειδί 2 με την μεταβλητή 'w'

Έξοδος: {1: 'a', 2: 'w', 3: 'c'}

5. a_dict .has_key(key)

Παράδειγμα:

```
print {1: 'a', 2: 'b', 3: 'c'}.has_key(2)
```

Ελέγχει εάν υπάρχει το κλειδί 2

Έξοδος: True

6. a_dict.pop(key)

Παράδειγμα:

```
a_dict = {1: 'a', 2: 'b', 3: 'c'}
print a_dict .pop(1)
print a_dict
```

Μετακινεί το κλειδί 1 από το a_dict

Έξοδος: a

```
{2: 'b', 3: 'c'}
```

7. a_dict.keys()

Παράδειγμα:

```
print {1: 'a', 2: 'b', 3: 'c'}.keys()
```

Επιστρέφει τα κλειδιά που υπάρχουν στο a_dict

Έξοδος:[1, 2, 3]

8 .a_dict.values()

Παράδειγμα:

```
print {1: 'a', 2: 'b', 3: 'c'}.values()
```

Επιστρέφει τις μεταβλητές που υπάρχουν στο a_dict

Έξοδος:['a', 'b', 'c']

6.3 ΦΟΡΤΩΣΗ ΕΙΚΟΝΑΣ ΑΠΟ ΤΟ INTERNET ΚΑΙ ΑΠΕΙΚΟΝΙΣΗ ΕΙΚΟΝΑΣ ΣΤΟΝ ΚΑΜΒΑ

ΣΥΝΤΑΞΗ:

simplegui.load_image(URL)

Η παραπάνω εντολή φορτώνει μία εικόνα απο το internet και δέχεται σαν παράμετρο το URL της εικόνας

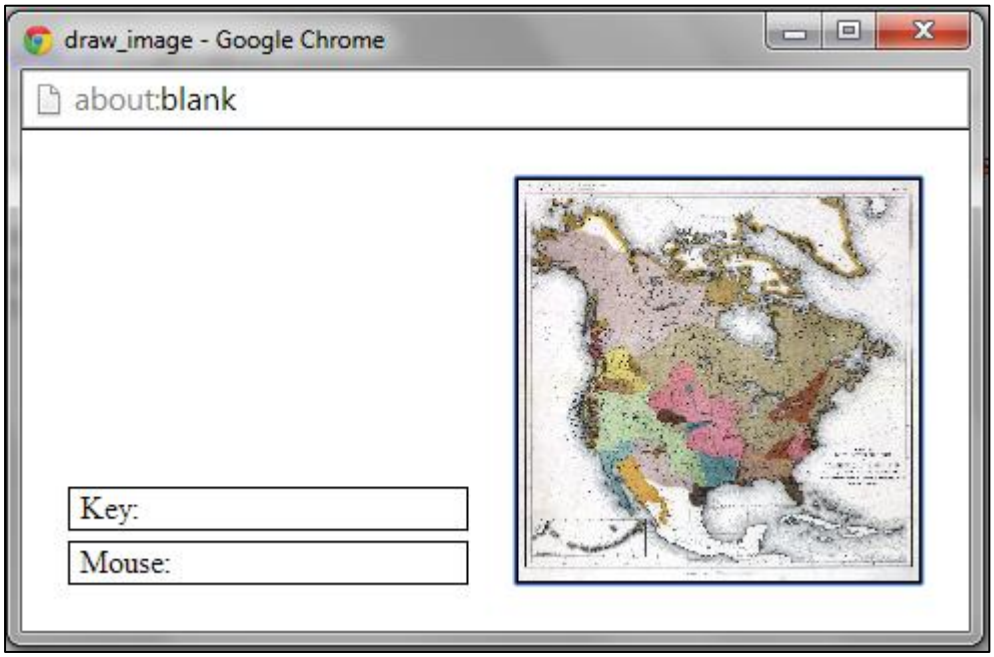
ΣΥΝΤΑΞΗ:

canvas.draw_image(image, center_source, width_height_source, center_dest, width_height_dest)

Η παράμετρος center_source είναι ένα ζευγάρι συντεταγμένων που δίνει την θέση που έχει το κέντρο της εικόνας ενώ το width_height_source είναι οι διαστάσεις της εικόνας.Το center_dest είναι ένα ζευγάρι συντεταγμένων που προσδιορίζει το κέντρο του καμβά όπου θα σχεδιαστεί η εικόνα και τέλος το width_height_dest είναι οι διαστάσεις του frame.Παρακάτω δίνεται ένα απλό παράδειγμα απεικόνισης εικόνας διαστάσεων 1521x1818 pixels.

```
import simplegui


def draw_handler(canvas):
    canvas.draw_image(image, (1521 / 2, 1818 / 2), (1521, 1818), (100, 100), (200, 200))


image = simplegui.load_image ('http://commondatastorage.googleapis.com/codeskulptor-assets/gutenberg.jpg')


frame = simplegui.create_frame('draw_image', 200, 200)

frame.set_draw_handler(draw_handler)

frame.start()
```

Έξοδος προγράμματος:



6.4 ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΑΣΚΗΣΗ

Την πέμπτη εβδομάδα του μαθήματος υλοποιήθηκε το παιχνίδι Memory.Το συγκεκριμένο παιχνίδι τεστάρει τις ικανότητες απομνημόνευσης του παίκτη.Πρέπει να δημιουργηθεί ένα frame στο οποίο θα απεικονίζονται δεκαέξι κάρτες με αριθμούς από το μηδέν έως το οχτώ. Αρχικά όλες οι κάρτες είναι γυρισμένες ανάποδα για να είναι κρυμμένες από τον παίκτη.Ο παίκτης έχει την δυνατότητα με την χρήση του ποντικιού να ανοίξει έως και δύο κάρτες. Ένας μετρητής(turn) θα καταχωρεί των αριθμό των προσπαθειών του παίκτη. Εάν οι ανοικτές κάρτες είναι ίδιες τότε παραμένουν ανοικτές ειδάλλως κλείνουν.Το παιχνίδι θα ολοκληρωθεί με επιτυχία όταν όλες οι κάρτες αποκαλυφθούν.

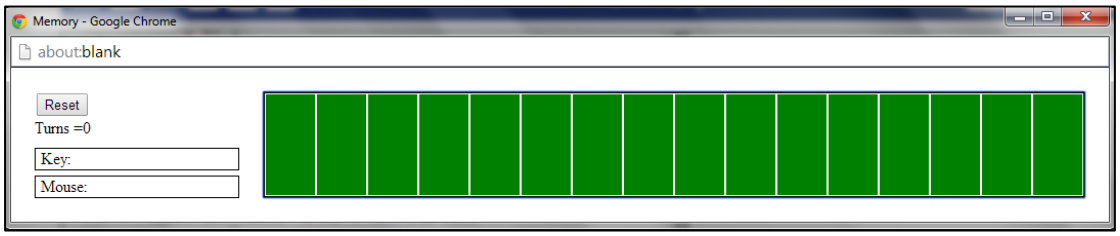
6.4.1 ΥΛΟΠΟΙΗΣΗ ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗΣ ΑΣΚΗΣΗΣ

```
Δήλωση βιβλιοθηκών simplegui και random
import simplegui
import random

Δήλωση  global μεταβλητών
card_width=800/16  #Το μήκος κάθε κάρτας ισούται με τον πλάτος του frame διαιρεμένο με το μήκος των καρτών
turns=0   #Στην μεταβλητή καταχωρούνται οι προσπάθειες του παίκτη και αρχικοποιείται σε μηδέν

Η συνάρτηση new_game εκκινεί το πρόγραμμα και αρχικοποιεί τις παραμέτρους του προγράμματος.
Οι μεταβλητές exposed, card, state και turns δηλώνονται σαν global για να χρησιμοποιηθούν οπουδήποτε στο πρόγραμμα.
def new_game():
    global exposed,card,state,turns
    state =0   #Η μεταβλητή state μετράει τον αριθμό των καρτών που είναι ανοικτές.
    turns=0  # Η μεταβλητή turns μετράει τις προσπάθειες του χρήστη, Η μεταβλητή turns αυξάνεται κάθε φορά που ανοίγουν δύο κάρτες
    card=[] #κενή λίστα στην οποία θα καταχωρηθούν οι κάρτες
    exposed = [False for i in range(16)] #Η exposed θέτει στις 16 κάρτες  τιμή False.Όσο οι κάρτες έχουν τιμή false τότε το περιεχόμενο τους μένει κρυμμένο από τον παίκτη


    #Δημιουργία λίστας num εύρους οχτώ στοιχείων με τιμές από το 0-7
    for num in range(8):
        card.append(num) # Στην κενή λίστα card προστίθενται τα  οχτώ στοιχεία της λίστας num με την εντολή append
        card.append(num) # Στη λίστα card που περιέχει οχτώ στοιχεία προστίθενται άλλα  οχτώ στοιχεία της λίστας num
        random.shuffle(card) #H  random.shuffle βάζει σε τυχαία σειρά τα στοιχεία της
```

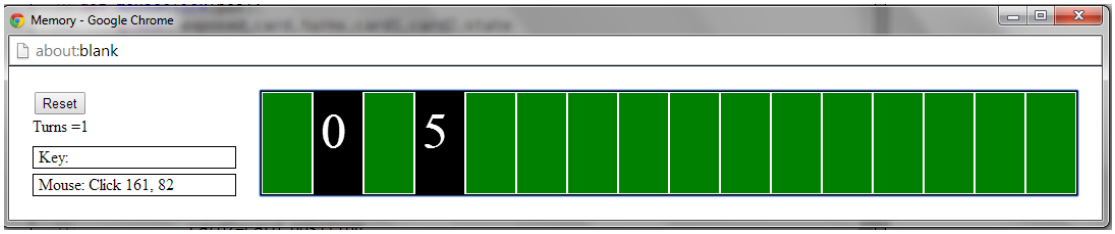
```
    λίστας card
    label.set_text('Turns='+str(turns)) # Με την εντολή label.set_text απεικονίζεται η
    μεταβλητή turns στην οθόνη

 Δήλωση συνάρτησης mouseclick η οποία ελέγχει την κατάσταση του παιχνιδιού
def mouseclick(pos):
  global exposed,card,turns,card1,card2,state

  card_position=pos[0]//card_width # Θέση κάθε κάρτας στον καμβά
  if state==0: #Εάν το state είναι μηδέν καμία κάρτα δεν είναι ανοικτή
      exposed[card_position]=True #Όταν ο παίκτης κάνει κλικ σε κάποια κάρτα του
  καμβά η κάρτα που βρίσκεται στην συγκεκριμένη θέση αποκαλύπτεται παίρνοντας
 τιμή True

      card1=card_position # Η τιμή  card_position εισάγεται στην μεταβλητή  card1με
την οποία δηλώνεται ότι μία κάρτα είναι ανοικτή
      state=1   # Το state γίνεται ένα άρα μία κάρτα είναι ανοικτή
Όταν το state έχει τιμή ένα και ο παίκτης κάνει κλικ σε μία δεύτερη κάρτα της οποίας η
θέση έχει τιμή False τότε αυτόματα παίρνει τιμή True και αποκαλύπτεται.
Όταν η τιμή του state παίρνει τιμή δύο τότε το turns αυξάνεται κατά ένα
  elif state==1:
      if exposed[card1]==True and exposed[card_position]==False:
        exposed[card_position]=True  # Η  δεύτερη κάρτα παίρνει τιμή True
        card2=card_position  # Η τιμή card_position εισάγεται στην μεταβλητή  card1
        state=2   #Το state είναι δύο άρα δύο κάρτες είναι ανοικτές
        turns= turns+1 #Αυξάνεται η τιμή του turns κατά ένα

Όταν το state είναι δύο τότε δύο κάρτες είναι ανοικτές.
  elif state == 2:
  Εάν οι  card1  και card2  είναι ανοικτές συγκρίνονται οι τιμές τους.
    if exposed[card1] == True and exposed[card2] == True :
  Εάν το περιεχόμενο των card1 και card2 είναι διαφορετικό τίθεται τιμή False στις
 card1 και card2 άρα οι κάρτες κλείνουν
      if card[card1] != card[card2]:
          exposed[card1] = False
          exposed[card2] = False
Όταν ο χρήστης κάνει κλικ σε μία νέα κάρτα η τιμή της γίνεται True και αποκαλύπτεται.
Η θέση της κάρτας εκχωρείται στην μεταβλητή card1 και το state παίρνει τιμή ένα
          exposed[card_position] = True
          card1 = card_position
          state = 1
    Εάν το περιεχόμενο των card1 και card2  είναι ίδιο τότε οι κάρτες μένουν ανοικτές
και το state παίρνει τιμή ένα
      elif card[card1] == card[card2]:
          exposed[card1] = True
          exposed[card2] = True
          exposed[card_position] = True
          card1 = card_position
          state = 1
```

```
def draw(canvas):

    #σχεδιασμός δεκαέξι γραμμών στον καμβά για τον διαχωρισμό των καρτών
    for k in range(16):
        canvas.draw_line((k*card_width, 0), (k*card_width, 100), 2, "White")

    # σχεδιασμός κάθε στοιχείου(l) που περιέχεται στην λίστα card
    i= 0
    for l in card:
        canvas.draw_text(str(l), (i*card_width+8, 55), 50, "White")
        i += 1
    # σχεδιασμός  πράσινων πολύγωνων που αναπαριστούν τις κάρτες όταν αυτές είναι
    κλειστές
    i = 0
    point=50
    for l in card:
         if not exposed[i]:
         canvas.draw_polygon([[i * card_width, 0],[point, 0],[point, 100], [i *
card_width, 100]], 2,"White","Green")
        i += 1
        point+=50


  #σχεδιασμός του Turns στον καμβά
   label.set_text("Turns ="+str(turns))



 Δημιουργία frame με το όνομα Memory
frame = simplegui.create_frame("Memory", 800, 100)

Δημιουργία Reset button το οποίο καλεί την συνάρτηση new_game η οποία εκκινεί το
παιχνίδι
frame.add_button("Reset", new_game)

 Δημιουργία label το οποίο απεικονίζει το Turns στην οθόνη
label = frame.add_label("Turns = 0")

Δήλωση mouseclick_handler το οποίο καλεί την συνάρτηση mouseclick
frame.set_mouseclick_handler(mouseclick)

 Δήλωση mouseclick_handler το οποίο καλεί την συνάρτηση draw
frame.set_draw_handler(draw)

Καλείται η new_game για την εκκίνηση του παιχνιδιού
new_game()

Εκκίνηση frame
frame.start()
```

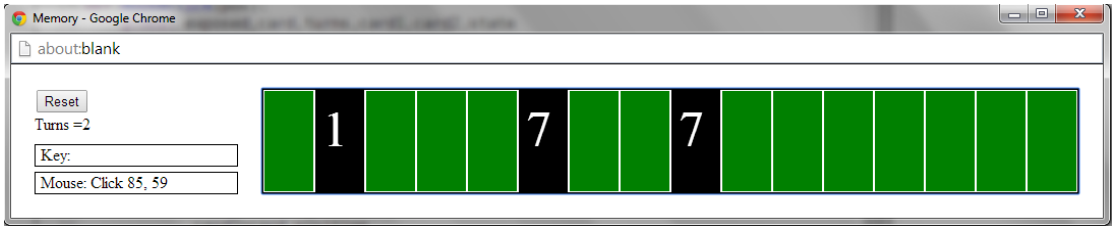
ΈΞΟΔΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ



Αφού ο χρήστης κάνει κλικ σε δύο κάρτες αυξάνεται η τιμή του Turns κατά 1.Εάν οι κάρτες δεν είναι ίδιες τότε θα κλείσουν αυτόματα όταν ο χρήστης θα προσπαθήσει να ανοίξει μια άλλη κάρτα.



Όταν ανοίξουν δύο κάρτες οι οποίες είναι ίδιες τότε παραμένουν ανοικτές όταν ο χρήστης θα κάνει κλικ σε μία τρίτη κάρτα.



7. BLACKJACK

Στο κεφάλαιο αυτό παρουσιάζουμε την υλοποίηση της προγραμματιστικής άσκησης Blackjack.Για τις ανάγκες υλοποίησης της προγραμματιστικής άσκησης παρουσιάζονται στο συγκεκριμένο κεφάλαιο η έννοια του αντικειμενοστραφή προγραμματισμού και ο τρόπος σύνταξης των κλάσεων. Αναλυτικά δίνονται οι εντολές του Codeskulptor για απεικόνιση των καρτών του παιχνιδιού όπως και παραδείγματα κλάσεων για την υλοποίηση της προγραμματιστικής άσκησης.

7.1 ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ(Object-oriented programming)

Ο αντικειμενοστραφής προγραμματισμός είναι ένας τρόπος οργάνωσης των προγραμμάτων που γράφουμε και συνδέεται άμεσα με κλάσεις(CLASSES) και αντικείμενα(OBJECTS).Κλάση είναι μία αναπαράσταση αντικειμένων ,δηλαδή κάποιων τύπων δεδομένων και ορίζει ποιες ιδιότητες(attributes) και ποιες μεθόδους(methods) θα έχει ένα αντικείμενο.Ας δούμε αναλυτικά πως λειτουργεί μία κλάση.Παρακάτω δημιουργήσαμε μία κλάση Character και στην συνέχεια ορίζουμε δύο μεθόδους την init και την str .Αυτές οι μέθοδοι μοιάζουν με συναρτήσεις αλλά δεν θεωρούνται συναρτήσεις από την στιγμή που βρίσκονται μέσα στην κλάση.Η πρώτη μέθοδος παίρνει τρία ορίσματα self,name και initial_health τα οποία καταχωρούνται σε τρία αντικείμενα self.name,self.health και self.inventory στις οποίες καταχωρούνται οι τιμές των παραμέτρων τις init name initial_health ενώ στο τρίτο αντικείμενο καταχωρείται μία κενή λίστα.Με την μέθοδο str επιστρέφουμε αντικείμενα τύπου string.Οι επόμενες δύο μέθοδοι χαρακτηρίζουν τις ιδιότητες που θα έχουν τα αντικείμενα.

```
class Character

    def __init__(self , name, initial_health):
        self.name = name
        self.health = initial_health
        self.inventory = []

    def __str__(self):
        s  = "Name: " + self.name
        s  +=  " Health " + str (self.health)
        s  += " Inventory " + str (self.inventory)
        return s

    def grab(self, item):
        self.inventory.append(item)

    def get_health(self):
        return self.health
```

Με την example δημιουργούμε έναν χαρακτήρα με όνομα Bob και initial_health 20.Αυτές οι τιμές καταχωρούνται στην μεταβλητή me.

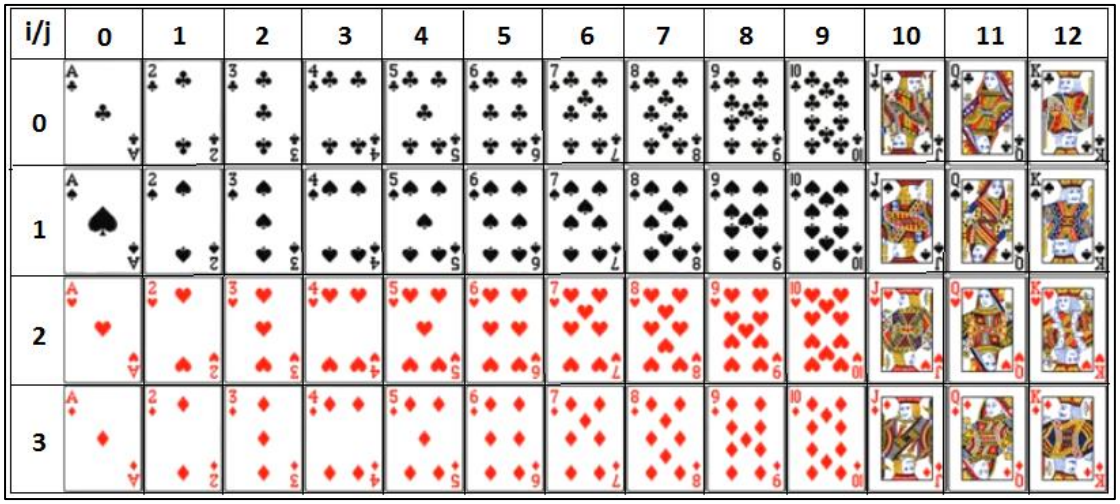
```
 def example():
      me=Character("Bob", 20)




Εκτυπώνεται η μεταβλητή me με την βοήθεια της μεθόδου str
      print str(me)

me.grab("pencil")
Με την μέθοδο grab προστίθεται στην λίστα self.inventory το “paper”
      me.grab("paper")

Εκτυπώνεται η μεταβλητή me με την βοήθεια της μεθόδου str
      print str(me)

Εκτυπώνεται η τιμή initial
      print "Health:" , me.get_health()

Καλείται η example
example()
```

Έξοδος προγράμματος

```
Name: Bob Health 20 Inventory []
Name: Bob Health 20 Inventory ['pencil', 'paper']
Health: 20
```

7.2 ΑΠΕΙΚΟΝΙΣΗ ΕΙΚΟΝΩΝ ΓΙΑ ΤΟ BLACKJACK

Η παρακάτω εικόνα αποτελείται από 52 κάρτες που το μέγεθος της κάθε κάρτας είναι CARD_SIZE(73, 98)και το κέντρο της είναι CARD_CENTER(36.5, 49) .Για να απεικονίσουμε την κάθε κάρτα στον καμβά θα πρέπει υπολογίσουμε την θέση της.Υποθέτουμε ότι οι κάρτες μας βρίσκονται σε ένα πίνακα τεσσάρων γραμμών και δεκατριών στήλες όπως φαίνεται παρακάτω.



Η θέση κάθε κάρτας ορίζεται ως εξής:

ΣΥΝΤΑΞΗ:

```
CARD_POS=(CARD_CENTER[0] + i * CARD_SIZE[0],
                CARD_CENTER[1] + j * CARD_SIZE[1])
```

Απεικόνιση κάρτας:

ΣΥΝΤΑΞΗ:

```
canvas.draw_image(όνομα εικόνας, card_pos, card_size, …)
```

7.2.1 ΠΑΡΑΔΕΙΓΜΑ ΑΠΕΙΚΟΝΙΣΗΣ ΚΑΡΤΩΝ

Δήλωση simplegui βιβλιοθήκης

```
import simplegui
```

Δηλώνουμε τις global μεταβλητές για τις κάρτες.Η μεταβλητή RANKS απεικονίζει τις κάρτες που βρίσκονται στον προηγούμενο πίνακα σε σειρά ταξινόμησης ενώ η μεταβλητή SUITS απεικονίζει τον αριθμό των γραμμών του πίνακα σε σειρά ταξινόμησης.

```
  RANKS = ('A', '2', '3', '4', '5', '6', '7', '8', '9', 'T', 'J', 'Q', 'K')
  SUITS = ('C', 'S', 'H', 'D')
```

Δηλώνουμε το μέγεθος και το κέντρο της κάθε κάρτας.

```
 CARD_SIZE = (73, 98)
 CARD_CENTER = (36.5, 49)
```

Δήλωση της διεύθυνσης URL που απεικονίζει τις κάρτες του blackjack

```
Card_images=simplegui.load_image(
        "http://commondatastorage.googleapis.com/codeskulptor-
assets/cards.jfitz.png")
```

Δημιουργία της κλάσης Card και μεθόδων init και draw για τον υπολογισμό της θέσης της κάθε κάρτας και την απεικόνιση της στον καμβά.

```
class Card:
    def __init__(self, suit, rank):
        self.suit = suit
        self.rank = rank
    def draw(self, canvas, loc):
         i = RANKS.index(self.rank)
         j = SUITS.index(self.suit)
         card_pos = [CARD_CENTER[0] + i * CARD_SIZE[0],
                     CARD_CENTER[1] + j * CARD_SIZE[1]]
         canvas.draw_image(card_image, card_pos ,CARD_SIZE,loc, CARD_SIZE)
```

Δημιουργία συνάρτησης που εμφανίζει μία κάρτα στο κέντρο του καμβά.

```
def draw(canvas):
     one_card.draw(canvas, (155, 90))
```

Δημιουργία frame.

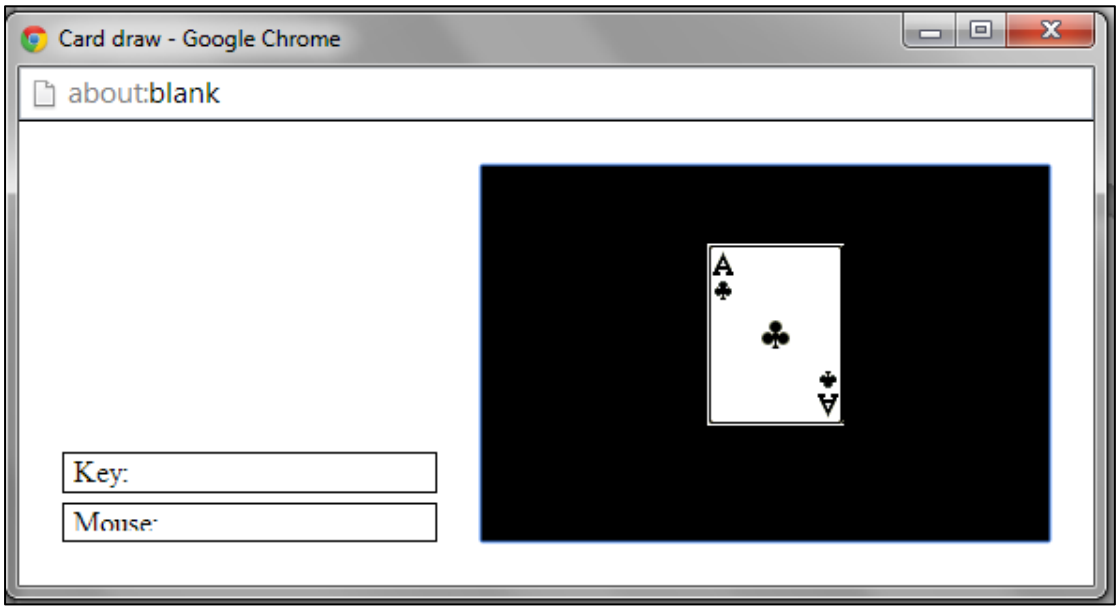
```
 frame = simplegui.create_frame("Card draw, 300, 200")
```

Δημιουργία draw handler με το οποίο καλείται η συνάρτηση draw και εμφανίζεται στην οθόνη ο άσσος που βρίσκεται στην θέση 'C' 'A' δηλαδή την θέση 0 του πίνακα.

```
frame.set_draw_handler(draw)
  one_card = Card('C', 'A')
```

```
Εκκίνιση frame
frame.start()
```

ΕΞΟΔΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ



7.3 ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΑΣΚΗΣΗ

Το blackjack είναι ένα τυχερό παιχνίδι γνωστό επίσης και εικοσιένα.Οι παίκτες του blackjack ανταγωνίζονται ενάντια στον κρουπιέρη και όχι ενάντια σε άλλους παίκτες Ό σκοπός του παιχνιδιού είναι ο παίκτης να πάρει μεγαλύτερο χέρι από το χέρι του κρουπιέρη αλλά να μην ξεπερνάει το συνολικό ποσό εικοσιένα.Αν ο παίκτης ξεπεράσει το εικοσιένα τότε «καίγεται» και χάνει την παρτίδα ανεξάρτητα από το χέρι του κρουπιέρη.

```
Δήλωση βιβλιοθηκών random και simplegui
import simplegui
import random

Δηλώνουμε το μέγεθος και το κέντρο της κάθε κάρτας.
CARD_SIZE = (73, 98)
CARD_CENTER = (36.5, 49)

Δήλωση της διεύθυνσης URL που απεικονίζει τις κάρτες του blackjack
Card_images=simplegui.load_image(
"http://commondatastorage.googleapis.com/codeskulptor-assets/cards.jfitz.png")


Δηλώνουμε το μέγεθος και το κέντρο του πίσω μέρους της κάθε κάρτας όπως και της
διεύθυνση URL που απεικονίζει το πίσω μέρος της κάθε κάρτας.
CARD_BACK_SIZE = (71, 96)
CARD_BACK_CENTER = (35.5, 48)
  card_back = simplegui.load_image
              ( "http://commondatastorage.googleapis.com/codeskulptor-
assets/card_back.png")

Αρχικοποίηση global μεταβλητών
in_play = False   #Η μεταβλητή  in_play έχει τιμή  False όταν το παιχνίδι δεν
βρίσκεται σε εξέλιξη
outcome = "" # Το outcome είναι μία μεταβλητή με την οποία εμφανίζονται
μηνύματα στον καμβά για τον χρήστη όσο βρίσκεται σε εξέλιξη ένα παιχνίδι
  score = 0 #Το σκορ του παιχνιδιού αρχικοποιείται σε μηδέν

Δηλώνουμε τις global μεταβλητές για τις κάρτες.Η μεταβλητή RANKS απεικονίζει τις
κάρτες που βρίσκονται στον προηγούμενο πίνακα σε σειρά ταξινόμησης ενώ η
μεταβλητή SUITS απεικονίζει τον αριθμό των γραμμών του πίνακα σε σειρά
ταξινόμησης.
RANKS = ('A', '2', '3', '4', '5', '6', '7', '8', '9', 'T', 'J', 'Q', 'K')
SUITS = ('C', 'S', 'H', 'D')

Δηλώνουμε το dictionary VALUES το οποίο αντιστοιχεί τα στοιχεία της λίστας RANKS
σε αριθμούς.
VALUES = {'A':1, '2':2, '3':3, '4':4, '5':5, '6':6, '7':7, '8':8, '9':9, 'T':10, 'J':10, 'Q':10,
'K':10}
```

Δηλώνουμε την κλάση Card όπου με την μέθοδο init αρχικοποιούμε τις κάρτες ενώ η μέθοδος str επιστρέφει αντικείμενα τύπου string.Οι μέθοδοι get_rank και get_suit επιστρέφουν τις τιμές των rank και suit αντίστοιχα.

```
class Card:
    def __init__(self, suit, rank):
        if (suit in SUITS) and (rank in RANKS):
            self.suit = suit
            self.rank = rank
        else:
            self.suit = None
            self.rank = None
            print "Invalid card: ", suit, rank
    def __str__(self):
        return self.suit + self.rank

    def get_suit(self):
        return self.suit

    def get_rank(self):
        return self.rank
```

Η μέθοδος draw σχηματίζει τις κάρτες στον καμβά.

```
def draw(self, canvas, pos):
    card_loc = (CARD_CENTER[0] + CARD_SIZE[0] * RANKS.index(self.rank),
                CARD_CENTER[1] + CARD_SIZE[1] * SUITS.index(self.suit))
    canvas.draw_image(card_images, card_loc, CARD_SIZE,
    [pos[0] + CARD_CENTER[0], pos[1] + CARD_CENTER[1]], CARD_SIZE)
```

Στην συνέχεια δημιουργούμε την κλάση Hand δηλαδή το χέρι του παίκτη και του κρουπιέρη στην οποία περιέχονται η μέθοδος init η οποία δημιουργεί μια κενή λίστα self.hand.Η μέθοδος str επιστρέφει όλα τα δεδομένα που βρίσκονται στην λίστα self.hand.

```
class Hand:
        def __init__(self):
        self.hand= []

        def __str__(self):
            string=''
            for x in self.hand:
            string =  string + str(x) + ' '
            return 'Hand contains'+ ' ' + string #Με το return επιστρέφεται το
περιεχόμενο του self.hand
```

Η μέθοδος add_card με την εντολή append προσθέτει κάρτες στην λίστα self.hand και επιστρέφει το αποτέλεσμα.

```
def add_card(self, card):
        self.hand.append(card)
        return self.hand
```

Με την μέθοδο get_value υπολογίζουμε τα χαρτιά που επιλέγει ο παίκτης και ο ντύλερ.Αρχικα η μεταβλητή value αρχικοποιείται σε μηδέν.Με την βοήθεια της μεθόδου get_rank της κλάσης Card παίρνουμε την τιμή κάθε κάρτας που επιλέγεται και την εισχωρούμε στην μεταβλητή value.

```
def get_value(self):
      value=0
       rank=''
       for card in self.hand:
            rank = card.get_rank()
            value = value + VALUES[rank]
       if  not ('A' in rank): #Εάν δεν υπάρχει άσσος τότε επιστρέφεται η τιμή value
            return value
       else:
          if value+10 <= 21:# Εάν υπάρχει άσσος και η τιμή value+10<=21τότε
          επιστρέφεται η τιμή value+10
      return  value+10
           else:
              return value# Εάν υπάρχει άσσος και η τιμή value+10>21 τότε επιστρέφεται
              η τιμή value
```

Κανονικά η τιμή του άσσου ισούται με ένα,αλλά στην περίπτωση που υπάρχει άσσος και η τιμή του χεριού δεν ξεπερνάει το εικοσιένα τότε δίνεται τιμή δέκα στον άσσο.

Η μέθοδος draw σχηματίζει τις κάρτες της τράπουλας στον καμβά.Οσο ένα παιχνίδι βρίσκεται σε εξέλιξη τότε οι κάρτες που βρίσκονται στο σημείο [135,248] του καμβά θα είναι κρυμένες.Στο σημείο [135,248] ορίσαμε τις κάρτες του ντίλερ.

```
def draw(self, canvas, pos):
        for card in self.hand:
           card.draw(canvas, pos)
           pos[0]=pos[0] + 100
         if in_play == True:
         canvas.draw_image(card_back, CARD_BACK_CENTER,
           CARD_BACK_SIZE, [135,248], CARD_BACK_SIZE)
```

Στην συνέχεια δημιουργούμε την κλάση Deck για την τράπουλα η οποία περιέχει τέσσερις μεθόδους

Η μέθοδος init απεικονίζει τις κάρτες

```
class Deck:
    def __init__(self):
        self.Deck = []
        for suit in SUITS:
           for rank in RANKS:
              self.Deck.append(Card(suit,rank))
```

Η μέθοδος shuffle ανακατεύει τις κάρτες

```
    def shuffle(self):
        random.shuffle(self.Deck)
```

Η deal_card παίρνει μία κάρτα από την τράπουλα

```
   def deal_card(self):
        return self.Deck.pop()
```

Η str εμφανίζει στην οθόνη το περιεχόμενο των καρτών

```
   def __str__(self):
        string = ''
        for x in self.Deck:
            string = string + str(x) + ' '
        return 'Deck contains'+ '' +string
```

Η deal καλείται όταν ο παίκτης πατήσει το button deal
Ανακατεύει από την αρχή την τράπουλα και εμφανίζει μήνυμα στην οθόνη ότι ο παίκτης έχασε την προηγούμενη παρτίδα

```
def deal():
    global outcome, in_play, score
    global deck, dealer_hand,player_hand

    if in_play==True: #Το παιχνίδι βρίσκεται σε εξέλιξη
    outcome='Player loses'
    deck=Deck()
    dealer_hand=Hand() #δημιουργία χαρτιών για τον ντίλερ
    player_hand=Hand() # δημιουργία χαρτιών για τον παίκτη
   deck.shuffle() #Η συνάρτηση shuffle ανακατεύει την τράπουλα


#Κώδικας για τον παίκτη
```

Καλείται η μέθοδος deal_card της κλάσης Deck η οποία παίρνει μία κάρτα από την τράπουλα και η τιμή της εισχωρείτε στην μεταβλητή card1

```
card1=deck.deal_card()
```

Η κάρτα card1 προστίθεται στο χέρι του παίκτη με την μέθοδο add_card της κλάσης Hand

```
player_hand.add_card(card1)
```

Καλείται η μέθοδος deal_card της κλάσης Deck η οποία παίρνει την δεύτερη κάρτα από την τράπουλα και εισχωρεί την τιμή της στην μεταβλητή card1

```
   card2=deck.deal_card()
```

Η κάρτα card1 προστίθεται στο χέρι του παίκτη με την μέθοδο add_card της κλάσης Hand

```
    player_hand.add_card(card2)


#κώδικας για τον κρουπιέρη
```

Καλείται η μέθοδος deal_card της κλάσης Deck η οποία παίρνει μία κάρτα από την τράπουλα και η τιμή της εισχωρείτε στην μεταβλητή card3.

```
       card3=deck.deal_card()
```

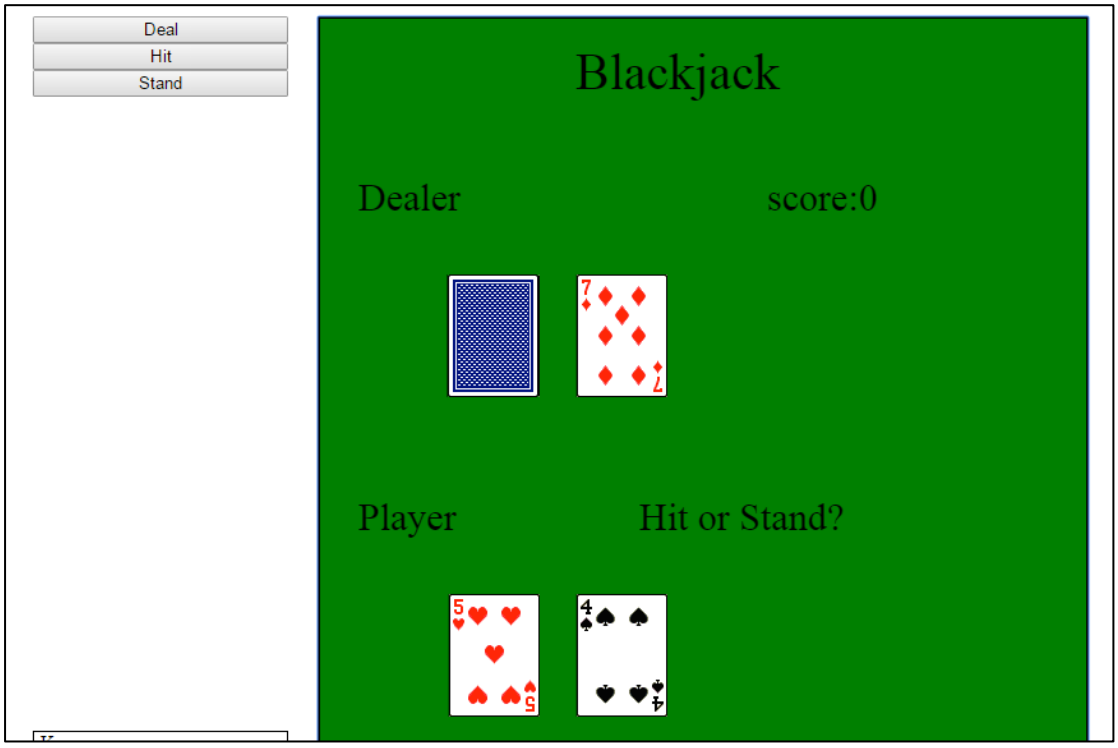
Η κάρτα card3 προστίθεται στο χέρι του κρουπιέρη με την μέθοδο add_card της

```
κλάσης Hand.
    dealer_hand.add_card(card3)

Καλείται η μέθοδος deal_card της κλάσης Deck  η οποία παίρνει μία κάρτα από την
τράπουλα και η τιμή της εισχωρείτε στην μεταβλητή card4.
    card4=deck.deal_card()
Η κάρτα card4 προστίθεται στο χέρι του κρουπιέρη με την μέθοδο add_card της
κλάσης Hand
    dealer_hand.add_card(card4)
    outcome='Hit or Stand?'
    score-=1 #To score μειώνεται όταν ο παίκτης πατήσει deal ενώ βρίσκεται σε
    εξέλιξη ένα παιχνίδι.

elif in_play==False:   # Εάν το παιχνίδι δεν βρίσκεται σε εξέλιξη
     deck =Deck ()  # Δημιουργείται η τράπουλα.
     dealer_hand=Hand () # Δημιουργείται το χέρι του κρουπιέρη.
    player_hand= Hand () # Δημιουργείται το χέρι του παίκτη.
    deck.shuffle () # Η εντολή shuffle ανακατεύει την τράπουλα.

#κώδικας για τον σχηματισμό της τράπουλας πριν την εκκίνηση του παιχνιδιού
#Κώδικας για τον παίκτη
Καλείται η μέθοδος deal_card της κλάσης Deck  η οποία παίρνει μία κάρτα από την
τράπουλα και η τιμή της εισχωρείτε στην μεταβλητή card1
     card1=deck.deal_card()

 Η κάρτα card1 προστίθεται στο χέρι του παίκτη με την μέθοδο add_card της κλάσης
Hand.
     player_hand.add_card(card1)

Καλείται η μέθοδος deal_card της κλάσης Deck  η οποία παίρνει μία κάρτα από την
τράπουλα και η τιμή της εισχωρείτε στην μεταβλητή card2
     card2=deck.deal_card()

Η κάρτα card1 προστίθεται στο χέρι του παίκτη με την μέθοδο add_card της κλάσης
Hand.
      player_hand.add_card(card2)

#κώδικας για τον κρουπιέρη
Καλείται η μέθοδος deal_card της κλάσης Deck  η οποία παίρνει μία κάρτα από την
τράπουλα και η τιμή της εισχωρείτε στην μεταβλητή card3
     card3=deck.deal_card()

Η κάρτα card3 προστίθεται στο χέρι του παίκτη με την μέθοδο add_card της κλάσης
Hand.
     dealer_hand.add_card(card3)

Καλείται η μέθοδος deal_card της κλάσης Deck  η οποία παίρνει μία κάρτα από την
τράπουλα και η τιμή της εισχωρείτε στην μεταβλητή card4.
     card4=deck.deal_card()
```

```
Η κάρτα card4προστίθεται στο χέρι του κρουπιέρη με την μέθοδο add_card της κλάσης Hand.
        dealer_hand.add_card(card4)
        outcome='Hit or Stand?'
        in_play=True


#Η συνάρτηση hit δίνει τις κάρτες στον παίκτη.
def hit():
    global deck,dealer_hand,player_hand,in_play,outcome, score

  if in_play: #Το παιχνίδι βρίσκεται σε εξέλιξη.
      #Η συνάρτηση hit δίνει κάρτες στον παίκτη καλώντας  την μέθοδο deal_card.
       card5=deck.deal_card()

       #Η κάρτα card4προστίθεται στο χέρι του παίκτη με την μέθοδο add_card της κλάσης Hand.
       player_hand.add_card(card5)

      #Εάν το χέρι του παίκτη είναι μεγαλύτερο του 21 τότε εμφανίζεται μήνυμα «You have busted»
       if player_hand.get_value()>21:
          outcome='You have busted.'+ 'Deal again'
           in_play=False # Το παιχνίδι σταματάει
           score-=1 # Το score μειώνεται κατά 1

def stand():
    global deck,dealer_hand,player_hand,in_play,outcome, score

    # Εάν το χέρι του κρουπιέρη είναι μικρότερο του 17 τότε δίνεται προστίθεται ακόμα μία κάρτα στο χέρι του.
    if in_play:
       while dealer_hand.get_value()<17:
           card5=deck.deal_card()
           dealer_hand.add_card(card5)

    # Εάν το χέρι του κρουπιέρη είναι μεγαλύτερο του 21  τότε ο κρουπιέρης χάνει.
       if dealer_hand.get_value()>21:
           outcome='Dealer is busted.'+'You won'
           score+=1 #Το score του παίκτη αυξάνεται κατά 1
           in_play=False #Το παιχνίδι σταματάει.

    # Εάν το χέρι του κρουπιέρη είναι μεγαλύτερο του χεριού του παίκτη τότε ο κρουπιέρης κερδίζει.
     elif dealer_hand.get_value()>player_hand.get_value():
           outcome='Dealer wins'
           score-=1 #Το score του παίκτη μειώνεται κατά 1
           in_play=False #Το παιχνίδι σταματάει
```

```
 # Εάν το χέρι του κρουπιέρη είναι μικρότερο του χεριού του παίκτη τότε ο
   παίκτης κερδίζει.
      elif dealer_hand.get_value()<player_hand.get_value():
         outcome='Dealer is busted.'+'You won'
         score+=1 #Το score του παίκτη αυξάνεται κατά 1
         in_play=False #Το παιχνίδι σταματάει

   # Εάν το χέρι του κρουπιέρη είναι ίσο με το χέρι του παίκτη τότε ο
   παίκτης κερδίζει.
      elif dealer_hand.get_value()==player_hand.get_value():
         outcome ='Dealer wins'
         score-=1 #Το score του παίκτη μειώνεται κατά 1
         in_play=False #Το παιχνίδι σταματάει

# Η συνάρτηση draw απεικονίζει το score τις κάρτες στον καμβά όπως επίσης και το
περιεχόμενο της μεταβλητής outcome .
def draw(canvas):
   global in_play,outcome

   canvas.draw_text("Blackjack",[200, 55], 40, "Black")
   canvas.draw_text("Dealer",[30,150],30,"Black")
   canvas.draw_text("Player",[30, 400],30,"Black")
   canvas.draw_text(outcome +",[250,400], 30,"Black")
   canvas.draw_text("score:"+str(score) ,[350,150], 30,"Black")
   dealer_hand.draw(canvas,[100,200])
   player_hand.draw(canvas,[100,450])


Δημιουργία frame και φόντου του frame σε χρώμα πράσινο.
frame = simplegui.create_frame("Blackjack", 600, 600)
frame.set_canvas_background("Green")

Δημιουργία buttons που καλούν τις συναρτήσεις deal,hit ,stand και draw
frame.add_button("Deal", deal, 200)
frame.add_button("Hit",  hit, 200)
frame.add_button("Stand", stand, 200)
frame.set_draw_handler(draw)

Καλείται η συνάρτηση deal που απεικονίζει τις κάρτες
deal()
Εκκίνηση  frame
frame.start()
```

ΈΞΟΔΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ



8.SPACESHIP

Στο κεφάλαιο αυτό παρουσιάζουμε την υλοποίηση της προγραμματιστικής άσκησης Spaceship.Αρχικά εξηγείται τι είναι ένα set και παρουσιάζονται και εξηγούνται οι παράμετροι της προγραμματιστικής άσκησης.

8.1 SETS

Ένα set είναι μία ακολουθία από μη ταξινομημένα στοιχεία στην οποία δεν επιτρέπεται η εισαγωγή ίδιων στοιχείων.

1.Δημιουργία set

```
name= set (['Kostas','Maria','Anna'])
print name
```

Έξοδος: set (['Kostas','Maria','Anna'])

```
name1= set (['Kostas','Maria','Anna','Eleni','Maria'])
print name1
```

Έξοδος: set (['Kostas','Maria','Anna','Eleni'])

Η έξοδος είναι η αναμενόμενη αφού δεν επιτρέπεται η εισαγωγή στοιχείου που ήδη βρίσκεται στο set

2.Επιλογή στοιχείου

```
name =set (['Kostas','Maria','Anna'])
for n in name:
    print n
```

Έξοδος:Kostas

Με την παραπάνω εντολή δεν επιλέγεται ένα συγκεκριμένο στοιχείο του set αλλά η επιλογή είναι τυχαία.

3. Προσθήκη στοιχείου

```
name = set (['Kostas','Maria','Anna'])
name.add ('Sofia')
```

Έξοδος: set (['Kostas','Maria','Anna','Sofia'])

3. Διαγραφή στοιχείου

```
name = set (['Kostas','Maria','Anna','Sofia']

name.remove ('Sofia')
```

Έξοδος: set (['Kostas','Maria','Anna'])

8.2 ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΑΣΚΗΣΗ

Για την συγκεκριμένη άσκηση πρέπει να δημιουργηθεί στην περιοχή του καμβά ένα διαστημόπλοιο,ένα γκρουπ μετεωριτών και ένα γκρουπ πυρών για το διαστημόπλοιο.Ο παίκτης με ένα keyboard handler θα ελέγχει την πορεία του διαστημοπλοίου.Το πρόγραμμα πρέπει κάθε δευτερόλεπτο να εμφανίζει ένα μετεωρίτη στον καμβά.Ο παίκτης δικαιούται τρείς ζωές.Εάν το διαστημόπλοιο συγκρουστεί με μετεωρίτες τότε ένας μετρητής μειώνει τις ζωές κατά ένα.Επίσης ο παίκτης μπορεί με την χρήση του πληκτρολογίου να χρησιμοποιήσει πυρά για να χτυπήσει μετεωρίτες. Εάν τα πυρά του διαστημοπλοίου έρθουν σε επαφή με μετεωρίτες τότε ο παίκτης αυξάνει το σκορ του κατά ένα.

8.2.1Αντικείμενα για την κλάση Ship

self.pos: Η θέση του διαστημοπλοίου που εκφράζεται σε ένα διάνυσμα ενός ζευγαριού τύπου float

self_vel: Η ταχύτητα του διαστημοπλοίου

self.angle: Η πορεία του διαστημοπλοίου

Το self.angle είναι η γωνία μεταξύ του οριζόντιου άξονα και της κατεύθυνσης του διαστημοπλοίου

self.angle_vel: Η γωνιακή ταχύτητα του διαστημοπλοίου.Η γωνιακή ταχύτητα είναι ένα διανυσματικό μέγεθος που εκφράζει την ταχύτητα ενός σώματος που εκτελεί κυκλική κίνηση.Είναι ένα διάνυσμα τύπου float.

Το self.angle_vel ελέγχει πόσο γρήγορα το διαστημόπλοιο περιστρέφεται όταν ο χρήστης χρησιμοποιεί το δεξί και το αριστερό βέλος του πληκτρολογίου.Ο έλεγχος του διαστημοπλοίου γίνεται με ένα key_handler.

self.thrust: Εάν το πλοίο επιταχύνει ή όχι.Εκφράζεται σε boolean

8.2.2 Ανανέωση θέσης διαστημοπλοίου

Για να κινηθεί το διαστημόπλοιο θα πρέπει να ανανεώνεται η θέση του .Κάτι τέτοιο επιτυγχάνεται εάν προστεθεί στην εκάστοτε θέση που έχει το διαστημόπλοιο η ταχύτητα του(self_vel).

```
self.pos [0] += self.vel[0]

self.pos [1] +=self.vel[1]
```

8.2.3 Επιτάχυνση του διαστημοπλοίου

Το διαστημόπλοιο θα πρέπει να επιταχύνει όταν ο χρήστης χρησιμοποιεί το πάνω βέλος του πληκτρολογίου.Έστω ότι η κόκκινη γραμμή του παρακάτω σχήματος εκφράζει το διάνυσμα της κατεύθυνσης του διαστημοπλοίου.Ο άξονας x είναι ο οριζόντιος άξονας ενώ ο άξονας y είναι ο κάθετος άξονας.Η γωνία που σχηματίζεται από τον άξονα x και την κατεύθυνση του διαστημοπλοίου είναι το self.angle.Ο άξονας x εκφράζεται ως το συνημίτονο του self.angle ενώ ο άξονας y το ημίτονο του self.angle.Εάν τα συνδυάσουμε αυτά τα δύο τότε υπολογίζεται το διάνυσμα της κατεύθυνσης του διαστημοπλοίου.

```
forward = [math.cos (self.angle),math.sin(self.angle)]
```

8.2.4 Ανανέωση ταχύτητας

Το τελευταίο που χρειάζεται η κλάση Ship είναι να ανανεωθεί η ταχύτητα του διαστημοπλοίου σε συνάρτηση με την κατεύθυνση forward.

```
If self.thrust:

    self.vel [0] += forward[0]

    self.vel [1] +=forward [1]
```

8.2.5 Friction (Τριβή)

Για να ελέγχεται πιο εύκολα το διαστημόπλοιο θα πρέπει να προστεθεί τριβή στην επιτάχυνση του διαστημοπλοίου. Δηλαδή όταν ο χρήστης χρησιμοποιεί το πάνω βέλος του πληκτρολογίου για να επιταχύνει το διαστημόπλοιο εάν το αφήσει τότε η τριβή θα κάνει το διαστημόπλοιο να σταματήσει. Ουσιαστικά η τριβή θα πρέπει να επιβραδύνει την ταχύτητα του διαστημοπλοίου και η επιβράδυνση αυτή να είναι μία σταθερά αντίθετης κατεύθυνσης από την ταχύτητα δηλαδή αρνητική.

```
Friction = -c * velocity
```

Τώρα το διαστημόπλοιο έχει αποκτήσει δύο επιταχύνσεις(acceleration) μία που του δίνει ώθηση(thrust) και μία που το επιβραδύνει(friction)

```
acceleration = thrust+friction
```

Άρα η ταχύτητα του διαστημοπλοίου θα είναι:

velocity = velocity +acceleration

velocity = velocity+thrust+friction

velocity = velocity+thrust-c*velocity

velocity = (1-c) * velocity+thrust

Η ταχύτητα που προκύπτει είναι η παλιά ταχύτητα του διαστημοπλοίου μειωμένη κατά μια σταθερά c. Εάν το διαστημόπλοιο επιταχύνει τότε στην ταχύτητα προστίθεται και η επιτάχυνση(thrust).

Ανανέωση τριβής

self.vel [0] *= (1-c)

self.vel [1] *= (1-c)

8.2.6 Συγκρούσεις αντικειμένων

Για να υπολογίσουμε εάν δύο αντικείμενα συγκρούονται είναι πιο εύκολο να υποθέσουμε ότι όλα τα αντικείμενα της άσκησης έχουν κυκλικό σχήμα ,άρα τότε όλα θα έχουν ένα κέντρο και μία ακτίνα.Εάν R1 και R2 η ακτίνα δύο αντικειμένων αντίστοιχα και D η απόσταση των δύο κέντρων τότε ισχύει.

1.Τα αντικείμενα δεν συγκρούονται όταν D>R1+R2

2. Τα αντικείμενα συγκρούονται όταν D<R1+R2

8.3 ΥΛΟΠΟΙΗΣΗ ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗΣ ΑΣΚΗΣΗΣ

```
# Δήλωση βιβλιοθηκών simplegui, math, random.
import simplegui
import math
import random

# Δήλωση global μεταβλητών.
WIDTH = 800
HEIGHT = 600
score = 0
lives = 3 #Η μεταβλητή lives δηλώνει τις ζωές που έχει ο παίκτης στην διάθεση του μέχρι να χάσει το παιχνίδι
time = 0.5 # Χρόνος για το κινούμενο φόντο του παιχνιδιού.
started = False # Η μεταβλητή started παίρνει τιμή False όταν δεν βρίσκεται σε εξέλιξη το παιχνίδι

Για να υλοποιηθεί το παιχνίδι Spaceship απαιτούνται αρκετές εικόνες οι οποίες αποτελούνται από πολλές παραμέτρους όπως μέγεθος εικόνας,μέγεθος κέντρου κτλ. Για να γίνουν διαχειρίσιμες  όλες αυτές οι πληροφορίες απαιτείται η δημιουργία μιας κλάσης. Η κλάση ImageInfo που δίνεται παρακάτω περιέχει πληροφορίες για το κέντρο,το μέγεθος,και την ακτίνα,την διάρκεια ζωής(lifespan) και το αν η εικόνα είναι
```

κινούμενη(animated) . Η ακτίνα στην περίπτωση του διαστημοπλοίου,είναι ένας κύκλος που το περιβάλει και είναι ιδιαίτερα χρήσιμος όταν θέλουμε να εντοπίσουμε συγκρούσεις με μετεωρίτες. Με αυτόν τον τρόπο δεν χρειάζεται να προβληματιζόμαστε για αντικείμενα που έχουν ιδιαίτερο σχήμα είναι αρκετά δύσκολο να υπολογιστεί το σημείο σύγκρουσής τους. Το lifespan είναι χρήσιμο για εικόνες που θέλουμε να εμφανίζονται για ένα ορισμένο χρονικό διάστημα όπως τα πυρά που εκτοξεύει το διαστημόπλοιο και το animated το χρησιμοποιούμε για εικόνες που θέλουμε να κινούνται χωρίς την παρέμβαση του χρήστη.

```
class ImageInfo:
    def __init__(self, center, size, radius = 0, lifespan = None, animated = False):
        self.center = center
        self.size = size
        self.radius = radius
        if lifespan:
            self.lifespan = lifespan
        else:
            self.lifespan = float('inf')
        self.animated = animated

     def get_center(self):
        return self.center

    def get_size(self):
        return self.size
        def get_center(self):
        return self.center

  def get_radius(self):
        return self.radius

    def get_lifespan(self):
        return self.lifespan

    def get_animated(self):
        return self.animated
```

Παρακάτω δίνονται όλες οι εικόνες που απαιτούνται για την υλοποίηση της άσκησης. Όλες οι εικόνες σχετίζονται άμεσα με την κλάση ImageInfo και δίνονται πληροφορίες για το κέντρο,το μέγεθος,την ακτίνα,την διάρκεια ζωής και εάν οι εικόνες είναι κινούμενες ή όχι. Με την εντολή simplegui.load_image φορτώνονται οι εικόνες από το αντίστοιχο URL

```
debris_info = ImageInfo([320, 240], [640, 480])
debris_image =
simplegui.load_image("http://commondatastorage.googleapis.com/codeskulptor-
assets/lathrop/debris2_blue.png")


nebula_info = ImageInfo([400, 300], [800, 600])
nebula_image =
simplegui.load_image("http://commondatastorage.googleapis.com/codeskulptor-
assets/lathrop/nebula_blue.f2014.png")


splash_info = ImageInfo([200, 150], [400, 300])
splash_image =
simplegui.load_image("http://commondatastorage.googleapis.com/codeskulptor-
assets/lathrop/splash.png")


ship_info = ImageInfo([45, 45], [90, 90], 35)
ship_image =
simplegui.load_image("http://commondatastorage.googleapis.com/codeskulptor-
assets/lathrop/double_ship.png")


missile_info = ImageInfo([5,5], [10, 10], 3, 50)
missile_image =
simplegui.load_image("http://commondatastorage.googleapis.com/codeskulptor-
assets/lathrop/shot2.png")


asteroid_info = ImageInfo([45, 45], [90, 90], 40)
asteroid_image =
simplegui.load_image("http://commondatastorage.googleapis.com/codeskulptor-
assets/lathrop/asteroid_blue.png")


explosion_info = ImageInfo([64, 64], [128, 128], 17, 24, True)
explosion_image =
simplegui.load_image("http://commondatastorage.googleapis.com/codeskulptor-
assets/lathrop/explosion_alpha.png")


soundtrack =
simplegui.load_sound("http://commondatastorage.googleapis.com/codeskulptor-
assets/sounddogs/soundtrack.mp3")

missile_sound =
simplegui.load_sound("http://commondatastorage.googleapis.com/codeskulptor-
assets/sounddogs/missile.mp3")
```

```
missile_sound.set_volume(.5)
ship_thrust_sound =
simplegui.load_sound("http://commondatastorage.googleapis.com/codeskulptor-assets/sounddogs/thrust.mp3")

explosion_sound =
simplegui.load_sound("http://commondatastorage.googleapis.com/codeskulptor-assets/sounddogs/explosion.mp3")
```

Η συνάρτηση angle_to_vector υπολογίζει το διάνυσμα της κατεύθυνσης του διαστημοπλοίου ενώ η συνάρτηση dist υπολογίζει την απόσταση δύο σημείων. Η συγκεκριμένη συνάρτηση είναι χρήσιμη όταν θέλουμε να υπολογίσουμε συγκρούσεις αντικειμένων.

```
def angle_to_vector(ang):
    return [math.cos(ang), math.sin(ang)]

def dist(p, q):
    return math.sqrt((p[0] - q[0]) ** 2 + (p[1] - q[1]) ** 2)
```

Η κλάση Ship περιέχει πληροφορίες για το διαστημόπλοιο. Η μέθοδος init αρχικοποιεί τις παραμέτρους του διαστημοπλοίου.
pos:Η θέση του διαστημοπλοίου
vel:Η ταχύτητα του διαστημοπλοίου
angle: Η κατεύθυνση του διαστημοπλοίου
image: Η εικόνα που απεικονίζει το διαστημόπλοιο. Συγκεκριμένα το URL για το διαστημόπλοιο που βρίσκεται στην κλάση ImageInfo περιέχει δύο εικόνες του διαστημοπλοίου,μία με τις μηχανές του διαστημοπλοίου σε κατάσταση ON και μία με μηχανές σε κατάσταση OFF.
Info: Περιέχει πληροφορίες για το διαστημόπλοιο. Με την βοήθεια των μεθόδων get_center, get_size, get_radius η κλάση Ship παίρνει τις τιμές για το κέντρο,το μέγεθος και την ακτίνα του διαστημοπλοίου που βρίσκονται στο ship_info της κλάσης ImageInfo
Το self.thrust σχετίζεται με την ώθηση του διαστημοπλοίου και αρχικά η τιμή του είναι False μέχρι ο χρήστης να εκκινήσει το διαστημόπλοιο.

```
class Ship:

    def __init__(self, pos, vel, angle, image, info):
        self.pos = [pos[0], pos[1]]
        self.vel = [vel[0], vel[1]]
        self.thrust = False
        self.angle = angle
        self.angle_vel = 0
        self.image = image
        self.image_center = info.get_center()
        self.image_size = info.get_size()
```

```
        self.radius = info.get_radius()
```
Η draw εμφανίζει το διαστημόπλοιο στον καμβά
```
  def draw(self,canvas):
        if self.thrust:
            canvas.draw_image(self.image, [self.image_center[0] + self.image_size[0],
self.image_center[1]] , self.image_size,
                              self.pos, self.image_size, self.angle)
        else:
            canvas.draw_image(self.image, self.image_center, self.image_size,
                              self.pos, self.image_size, self.angle)
```

Η get_position επιστρέφει την θέση του διαστημόπλοιου
```
    def get_position(self):
        return self.pos
```
Η get_radius επιστρέφει την ακτίνα του διαστημόπλοιου
```
    def get_radius(self):
        return self.radius
```
Η μέθοδος update ανανεώνει την θέση του διαστημόπλοιου
```
    def update(self):
         self.angle += self.angle_vel


        self.pos[0] = (self.pos[0] + self.vel[0]) % WIDTH
        self.pos[1] = (self.pos[1] + self.vel[1]) % HEIGHT

        #ανανέωση ταχύτητας
        if self.thrust:
            acc = angle_to_vector(self.angle)
            self.vel[0] += acc[0] * .1
            self.vel[1] += acc[1] * .1

        self.vel[0] *= .99
        self.vel[1] *= .99
```
#Η set_thrust ενεργοποιεί τους ήχους του διαστημοπλοίου όταν βρίσκεται σε κίνηση
```
    def set_thrust(self, on):
        self.thrust = on
        if on:
            ship_thrust_sound.rewind()
            ship_thrust_sound.play()
        else:
            ship_thrust_sound.pause()
```
#Αύξηση της γωνιακής ταχύτητας για την περιστροφή του διαστημοπλοίου
```
    def increment_angle_vel(self):
        self.angle_vel += .05
```
#Μείωση της γωνιακής ταχύτητας για την περιστροφή του διαστημοπλοίου
```
    def decrement_angle_vel(self):
        self.angle_vel -= .05
```

```
 #Η shoot είναι υπεύθυνη για την ρίψη πυρών στους μετεωρίτες
def shoot(self):
        global missile_group
        forward = angle_to_vector(self.angle)
        missile_pos = [self.pos[0] + self.radius * forward[0], self.pos[1] + self.radius *
forward[1]]
        missile_vel = [self.vel[0] + 8* forward[0], self.vel[1] + 8*  forward[1]]
        a_missile = Sprite(missile_pos, missile_vel, self.angle, 0, missile_image,
missile_info, missile_sound)
        missile_group.add(a_missile)

explosion_group = set([])

Κλάση για τον μετεωρίτη
class Sprite:
    def __init__(self, pos, vel, ang, ang_vel, image, info, sound = None):
        self.pos = [pos[0],pos[1]]
        self.vel = [vel[0],vel[1]]
        self.angle = ang
        self.angle_vel = ang_vel
        self.image = image
        self.image_center = info.get_center()
        self.image_size = info.get_size()
        self.radius = info.get_radius()
        self.lifespan = info.get_lifespan()
        self.animated = info.get_animated()
        self.age = 0
        if sound:
            sound.rewind()
            sound.play()
Η get_position επιστρέφει την θέση του μετεωρίτη
    def get_position(self):
        return self.pos
 Η get_radius επιστρέφει την ακτίνα του μετεωρίτη
    def get_radius(self):
        return self.radius
 Η draw εμφανίζει τους μετεωρίτες στον καμβά
    def draw(self, canvas):
        if self.animated == False:
            canvas.draw_image(self.image, self.image_center, self.image_size,
                        self.pos, self.image_size, self.angle)
        else:
            self.image_center[0]= 64 + (self.age*128)
            canvas.draw_image(self.image, self.image_center, self.image_size,self.pos,
self.image_size, self.angle)

Η μέθοδος update ανανεώνει την θέση του μετεωρίτη
    def update(self):

        self.angle += self.angle_vel
```

```
        self.pos[0] = (self.pos[0] + self.vel[0]) % WIDTH
        self.pos[1] = (self.pos[1] + self.vel[1]) % HEIGHT
        self.age += 1
        if self.age >= self.lifespan:
            return True
        else:
            return False
```

```
Οι συναρτήσεις key down και key up ελέγχουν το διαστημόπλοιο. Με τα  πλήκτρα 'left' και 'right' περιστρέφουμε το διαστημόπλοιο αριστερά και δεξιά με την βοήθεια των μεθόδων decrement_angle_vel και increment_angle_vel οι οποίες μειώνουν και αυξάνουν την γωνιακή ταχύτητα αντίστοιχα. Το πλήκτρο 'up'  επιταχύνει το διαστημόπλοιο  θέτοντας το self.thrust True και το space μέσω της συνάρτησης shoot() εκτοξεύει πυρά από το διαστημόπλοιο.
def keydown(key):
    if key == simplegui.KEY_MAP['left']:
        my_ship.decrement_angle_vel()
    elif key == simplegui.KEY_MAP['right']:
        my_ship.increment_angle_vel()
    elif key == simplegui.KEY_MAP['up']:
        my_ship.set_thrust(True)
    elif key == simplegui.KEY_MAP['space']:
        my_ship.shoot()


def keyup(key):
    if key == simplegui.KEY_MAP['left']:
        my_ship.increment_angle_vel()
    elif key == simplegui.KEY_MAP['right']:
        my_ship.decrement_angle_vel()
    elif key == simplegui.KEY_MAP['up']:
        my_ship.set_thrust(False)


#Η μέθοδος collide υπολογίζει συγκρούσεις μεταξύ δύο αντικειμένων
def collide(self, other_object):

 #Η μέθοδος get_position και get_radius δίνει την θέση και την ακτίνα του πρώτου αντικειμένου
        first_object_pos = self.get_position()
        first_object_radius = self.get_radius()
```

```
#Η μέθοδος get_position και get_radius δίνει την θέση και την ακτίνα του δεύτερου αντικειμένου
        second_object_pos = other_object.get_position()
        second_object_radius = other_object.get_radius()

#Υπολογίζεται η απόσταση των δύο αντικειμένων με την χρήση της συνάρτησης dist
        d = dist(first_object_pos, second_object_pos)


#Εάν η απόσταση μεταξύ τους είναι μεγαλύτερη ή ίση του αθροίσματος των δύο ακτίνων τότε επιστρέφει True αλλιώς επιστρέφει False
        if d <= (first_object_radius + second_object_radius):
            return True
        else:
            return False


#Η συνάρτηση rock_spawner δημιουργεί μία ομάδα μετεωριτών(rock_group).Αρχικά το rock_group δηλώνεται σαν ένα κενό set. Η συνάρτηση rock_spawner καλείται από ένα time_handler και κάθε ένα δευτερόλεπτο δημιουργεί έναν μετεωρίτη.
def rock_spawner():
    global rock_group

#Με την συνάρτηση random.randrange δίνονται τυχαίες τιμές για την θέση,την ταχύτητα και την ταχύτητα περιστροφής των μετεωριτών.
    rock_pos = [random.randrange(0, WIDTH), random.randrange(0, HEIGHT)]
    rock_vel = [random.random() * 0.5, random.random() * 0.5]
    rock_ang_vel = random.random() *0.1


#Δημιουργία ενός μετεωρίτης a_rock καλώντας την κλάση Sprite
    a_rock = Sprite(rock_pos, rock_vel, 0, rock_ang_vel, asteroid_image, asteroid_info)


#Η if ελέγχει τον αριθμό μετεωριτών του rock_group. Εάν ο αριθμός είναι μικρότερος του 12 τότε προστίθεται ένας μετεωρίτης
    if len(rock_group)<=12:
        rock_group.add(a_rock)

rock_group = set([])
timer = simplegui.create_timer(1000, rock_spawner)

#Η συνάρτηση group_collide ελέγχει συγκρούσεις μετεωριτών με αντικείμενα
def group_collide(sprite_group, other_object):

#Δημιουργούμε ένα set(set_of_sprite) που περιλαμβάνει την ομάδα των μετεωριτών και μία κενή λίστα(list)
    set_of_sprite = set(sprite_group)
    list = []
```

```
# Για κάθε αντικείμενο(x) που περιλαμβάνεται στο set_of_sprite γίνεται έλεγχος για σύγκρουση μέσω της συνάρτησης collide
    for x in set_of_sprite:
        if x.collide(other_object):

#Το αντικείμενο αφαιρείται από το spite_group και προστίθεται στην κενή λίστα
            sprite_group.remove(x)
            list.append(x)

#Το μήκος της λίστας επιστρέφεται έτσι ώστε να είναι γνωστός ο αριθμός των συγκρούσεων
    return len(list)


#Η συνάρτηση group_group_collide ελέγχει συγκρούσεις πυρών με μετεωρίτες
def group_group_collide(group_of_missiles, group_of_sprites):

#Δημιουργούμε ένα set(set_of_missiles)  που περιλαμβάνει την ομάδα των πυρών και μία κενή λίστα
    set_of_missiles = set(group_of_missiles)
    list1 = []

#Για κάθε αντικείμενο που βρίσκεται στο set_of_missiles γίνεται έλεγχος για σύγκρουση μέσω της συνάρτησης group_collide
    for m in set_of_missiles  :
        if group_collide(group_of_sprites, m):

#Το αντικείμενο αφαιρείται από το group_of_missiles και προστίθεται στην κενή λίστα
            group_of_missiles.remove(m)
            list1.append(m)

#Το μήκος της λίστας επιστρέφεται έτσι ώστε να είναι γνωστός ο αριθμός των συγκρούσεων.
    return len(list1)

#Η συνάρτηση process_group καλεί τη μεθόδους draw και update για κάθε μετεωρίτη.
#Η συνάρτηση παίρνει σαν παραμέτρους ένα set που και έναν καμβά.
def process_sprite_group(set_sprite, canvas):

   Δημιουργούμε ένα set(α)που περιλαμβάνει το set των μετεωριτων
    a = set(set_sprite)

#Για κάθε μετεωρίτη που βρίσκεται στο set(a)  καλούνται οι μέθοδοι draw και update
    for sprite in a:
        spite.draw(canvas)
        spite.update()

#Κάθε φορά που καλείται το update  τότε θα αφαιρείται  ο μετεωρίτης από set_sprite.
```

```
        if spite.update()
            set_sprite.remove(sprite)

#Συνάρτηση draw η οποία σχεδιάζει τα αντικείμενα στον καμβά
def draw(canvas):
    global time, started, process_sprite_group, lives, score, my_ship, soundtrack

    # κώδικας για το κινούμενο φόντο του παιχνιδιού.

    time += 1

   center = debris_info.get_center()
    size = debris_info.get_size()
    wtime = (time / 8) % center[0]
    canvas.draw_image(nebula_image, nebula_info.get_center(), nebula_info.get_size(),
[WIDTH / 2, HEIGHT / 2], [WIDTH, HEIGHT])
    canvas.draw_image(debris_image, [center[0] - wtime, center[1]], [size[0] - 2 *
wtime, size[1]],
                      [WIDTH / 2 + 1.25 * wtime, HEIGHT / 2], [WIDTH - 2.5 *
wtime, HEIGHT])
    canvas.draw_image(debris_image, [size[0] - wtime, center[1]], [2 * wtime, size[1]],
                      [1.25 * wtime, HEIGHT / 2], [2.5 * wtime, HEIGHT])

    #απεικόνιση του Score και του Lives στον καμβά
    canvas.draw_text("Lives", [50, 50], 22, "White")
    canvas.draw_text("Score", [680, 50], 22, "White")
    canvas.draw_text(str(lives), [50, 80], 22, "White")
    canvas.draw_text(str(score), [680, 80], 22, "White")



#απεικόνιση  του διαστημοπλοίου στον καμβά
     my_ship.draw(canvas)

#Καλείται η συνάρτηση process_sprite_group η οποία καλεί τη μεθόδους draw και
update για κάθε μετεωρίτη
    process_sprite_group(rock_group, canvas)

#Γίνεται έλεγχος  για σύγκρουση του διαστημοπλοίου και των μετεωριτών μέσω της
συνάρτησης group_collide.
#Εάν συγκρούονται τα δύο αντικείμενα τότε μειώνεται ο μετρητής της μεταβλητής  lives
κατά ένα.
    if group_collide(rock_group, my_ship):
        lives -= 1

#Καλείται η συνάρτηση process_sprite_group η οποία καλεί τη μεθόδους draw και
update για κάθε μετεωρίτη
    process_sprite_group(missile_group, canvas)

#Γίνεται έλεγχος  για σύγκρουση του γκρουπ των μετεωριτών και του γκρουπ των πυρών
```

```
μέσω της συνάρτησης group_group_collide.
#Εάν συγκρούονται τα δύο αντικείμενα τότε αυξάνεται ο μετρητής του score κατά ένα.
    if group_group_collide(rock_group, missile_group):
        score += 1

#Όταν ο μεταβλητή lives θα έχει τιμή μηδέν τότε το παιχνίδι σταματά μέχρι ο χρήστης να
ξεκινήσει καινούργιο παιχνίδι.
    if lives == 0:
        started = False
        lives = 3 # Όταν το παιχνίδι ξεκινήσει η μεταβλητή παίρνει τιμή τρία.

        my_ship.pos = [WIDTH/2, HEIGHT/2] #Η θέση εκκίνησης  του
        διαστημοπλοίου είναι το μέσο του καμβά.
  #Η for εξασφαλίζει ότι το rock_group του τελευταίου παιχνιδιού δεν απεικονίζεται
  στην οθόνη  με την έναρξη καινούργιου παιχνιδιού.
        for item in rock_group:
            rock_group.remove(item)

#Καλείται η update για να ανανεώσει την θέση του διαστημοπλοίου
    my_ship.update()

#Δημιουργία του frame
frame = simplegui.create_frame("Asteroids", WIDTH, HEIGHT)

# Δήλωση του διαστημοπλοίου,του γκρουπ των μετεωριτών και του γκρουπ των πυρών.
my_ship = Ship([WIDTH / 2, HEIGHT / 2], [0, 0], 0, ship_image, ship_info)
rock_group = set([])
missile_group = set([])

#Δήλωση των keyboard handlers για τον έλεγχο του διαστημοπλοίου μέσω του
πληκτρολογίου.
frame.set_keyup_handler(keyup)
frame.set_keydown_handler(keydown)
#Δήλωση draw_handler που καλεί την συνάρτηση draw.
frame.set_draw_handler(draw)

#Δήλωση χρονομέτρου που καλεί την συνάρτηση rock_spawner κάθε ένα δευτερόλεπτο.
timer = simplegui.create_timer(1000.0, rock_spawner)
 # Εκκίνηση χρονομέτρου και frame.
timer.start()
frame.start()
```

ΈΞΟΔΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ