ΑΕΙ ΠΕΙΡΑΙΑ Τ.Τ.
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ
ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ Τ.Ε.

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη Βιντεοπαιχνιδιών με Χρήση της Game Engine Unity

Ιωάννης Σπέντζας
Εισηγητής: Δρ Νίκος Ζ. Ζάχαρης, Καθηγητής

ΑΘΗΝΑ
ΑΠΡΙΛΙΟΣ 2015

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη Βιντεοπαιχνιδιών με Χρήση της Game Engine Unity

Ιωάννης Σπέντζας
Α.Μ. 39199

Εισηγητής: Δρ Νίκος Ζ. Ζάχαρης, Καθηγητής

Εξεταστική Επιτροπή

_____

_____

Ημερομηνία εξέτασης: _____

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία ολοκληρώθηκε μετά από μήνες δουλειάς πάνω σε ένα αντικείμενο που με ενδιαφέρει ιδιαίτερα. Την προσπάθειά μου αυτή υποστήριξε ο επιβλέπων καθηγητής μου, Νίκος Ζ. Ζάχαρης, τον οποίο θα ήθελα να ευχαριστήσω.

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία ασχολείται με την ανάπτυξη βιντεοπαιχνιδιών με χρήση της game engine Unity. Η ανάπτυξη βιντεοπαιχνιδιών είναι ένα από τα σημαντικότερα κομμάτια της βιομηχανίας της διασκέδασης και ένας πολύ προσιτός τομέας για νεαρούς προγραμματιστές οι οποίοι θέλουν να εργαστούν. Το παρακάτω κείμενο κάνει μια εισαγωγή στην ιστορία της βιομηχανίας των παιχνιδιών, το game development, τις game engine και δείχνει τον βασικό χειρισμό της μηχανής Unity, με σκοπό την δημιουργία ενός απλοϊκού παιχνιδιού

ABSTRACT

The present thesis concerns the development of videogames using the game engine Unity. Game development is one of the leading parts of the entertaiment industry, and a very approchable working possibility for young programmers. This thesis will cover the history of the video game industry, game development, game engines and it will show basic functions of Unity, in order for a game to be made with it.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Αρχιτεκτονική Ηλεκτρονικών Υπολογιστών

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: game engine, βιντεοπαιχνίδι, scripting, ανάπτυξη, unity, βιομηχανία διασκέδασης

ΠΕΡΙΕΧΟΜΕΝΑ

Κεφάλαιο 1: Η βιομηχανία της διασκέδασης και των παιχνιδιών

Κεφάλαιο 2: Ανάπτυξη βιντεοπαιχνιδιών

Κεφάλαιο 3: Παρουσίασή των βασικών λειτουργιών της μηχανής unity

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

Κεφάλαιο 1: Η βιομηχανία της διασκέδασης και των παιχνιδιών

1.1 Εισαγωγή στη βιομηχανία της διασκέδασης

Η βιομηχανία της διασκέδασης είναι η βιομηχανία η οποία περιλαμβάνει όλες τις υπηρεσίες και αγαθά διασκέδασης τα οποία έχουν κάποιο αντίκτυπο στην οικονομία. Μετά απο το 2000 η βιομηχανία της διασκέδασης είχε μια τρόμερή άνοδο στους περισότερους τομείς της, και μέχρι σήμερα έχει δημιουργήσει εκατομύρια θέσεις εργασίας.

Εικόνα 1.1 : Στατιστικά της βιομηχανίας

Ένα πολύ σημαντικό μέρος της βιομηχανίας της διασκέδασης είναι η βιομηχανία των βιντεοπαιχνιδιών. Χάρις την απλοποίηση της ανάπτυξης ανεξάρτητων

βιντεοπαιχνιδιών (δηλαδή χωρίς την βοήθεια κάποιου εκδότη) , την απλοποίηση αγοράς και πώλησης ψηφιακών παιχνιδιών μέσω πλατφόρμων όπως το Steam ή το Origin, και χάρις της "Nerd" κουλτούρας η οποία έγινε δημοφιλής στο ευρύ κοινό, η βιομηχανία των βιντεοπαιχνιδιών είδε μια πάρα πολύ γρήγορη άνοδο και αποδοχή από το ευρύ κοινό. Με 76 δις δολάρια έσοδα το 2013, έχει ξεπεράσει πολλές άλλες βιομηχανίες ψυχαγωγίας. Στην Αμερική έχει ξεπεράσει την βιομηχανία ταινιών από το 2007, και στην Αγγλία έχει ξεπεράσει την βιομηχανία μουσικής από το 2008.

Η βιομηχανία βιντεοπαιχνιδιών περιλαμβάνει χιλιάδες θέσεις εργασίας παγκόσμιος σε πολλούς κλάδους, από προγραμματισμό, σε τέχνες όπως μουσική ζωγραφική και 3d modeling, έως και θέσεις management και marketing ,καθώς και κλάδους που δημιουργήθηκαν συγκεκριμένα για τα βιντεοπαιχνίδια, όπως σχεδιαστής επιπέδων και σχεδιαστής gameplay. Χάρις την ψηφιακή μορφή της βιομηχανίας , υπάρχουν πολλές ευκαιρίες για δουλειά εξ-αποστάσεως, και αυτό δίνει την ευκαιρία σε ανθρώπους από χώρες στις οποίες η βιομηχανία δεν έχει αναπτυχθεί, να πιάσουν δουλειά στο εξωτερικό ποιο εύκολα. Επίσης χάρις της απλοποίησης χρήσης των μηχανών για ανάπτυξη παιχνιδιών (game engine), τον όλο και αυξάνων αριθμό documentation και tutorial, και την ανεξαρτητοποίηση των μικρών παιχνιδιών από τους εκδότες χάρις πλατφόρμες ψηφιακής πωλήσεως βιντεοπαιχνιδιών, η δημιουργία της ανεξάρτητης σκηνή ανάπτυξης βιντεοπαιχνιδιών έχει δώσει ευκαιρία σε πολλούς ερασιτέχνες να γίνουν επαγγελματίες , χωρίς να χρειαστεί να προσληφτούν από κάποια εταιρία ανάπτυξης βιντεοπαιχνιδιών. Τέλος τα τελευταία χρόνια έχουν εμφανιστεί τρόποι χρηματοδότησης παιχνιδιών από τους ίδιους τους χρήστες (crowd funding) όπως το Kickstarter και Indiegogo, δίνοντας την δυνατότητα σε μικρές ομάδες developer, να φτιάξουν βιντεοπαιχνίδια χωρίς να χρειάζεται να συμβιβαστούν με κάποιον εκδότη

Η βιομηχανία των βιντεοπαιχνιδιών είχε την πιο γρήγορη άνοδο από τις βιομηχανίες διασκέδασης, και θεωρείτε ότι μέσα στην επόμενη δεκαετία θα συνεχίσει την άνοδο και θα ξεπεράσει τα 100 δισ. σε ετήσια έσοδα

1.2 Ιστορική αναδρομή

1.2.1 1970

Εικόνα 1.2 : Το Magnavox Odyssey,βγήκε το 1972 και είναι η πρώτη κονσόλα για χρήση στο σπίτι.

Η βιομηχανία των βιντεοπαιχνιδιών θεωρείτε ότι ξεκίνησε το 1972, όταν η εταιρία Atari κυκλοφόρησε το παιχνίδι Pong, το πρώτο βιντεοπαιχνίδι που είχε εμπορική επιτυχία. Η πρώτη έκδοση του πώλησε πάνω από 19.000 μηχανές arcade. Εκείνη την χρονιά εμφανίστηκε η πρώτη κονσόλα για το σπίτι, το Magnavox Odyssey, αλλά επειδή κλώνοι του Pong σε φθηνές τιμές είχαν γεμίσει την αγορά, και χωρίς κάποιον ποιοτικό έλεγχο οι πελάτες έχασαν την εμπιστοσύνη τους, και αυτό οδήγησε στην πρώτη συντριβή της αγοράς το. Αυτή η συντριβή σταμάτησε με την επιτυχία του Space Invaders το 1978 , η οποία ξεκίνησε μια "χρυσή" εποχή για την βιομηχανία των βιντεοπαιχνιδιών . Αυτή η επιτυχία έκανε τις καμπίνες arcade να εμφανιστούν σε δημοφιλής περιοχές, όπως εμπορικά κέντρα και εστιατόρια. Το Space Invaders κατάφερε να πωλήσει πάνω από 360.000 μηχανές arcade, και μέχρι το 1982 είχε έσοδα 2 δις. δολάρια.

Μόλις παρατηρήθηκε η επιτυχία του, έγινε συμφωνία για την δημιουργία ενός version του Space Invaders για την κονσόλα Atari VCS (που αργότερα ονομάστηκε σε Atari 2600) , και έγινε η πρώτη επιτυχία της κονσόλας, τετραπλασιάζοντας τις πωλήσεις της. Η επιτυχία της κονσόλας Atari 2600 ξαναζωντάνεψε την αγορά των Home Console στην δεύτερη γενεά τους

1.2.2 1980

Εικόνα 1.3 :Το Nintendo Entertainment System, βγήκε το 1985, και βόηθησε την αμερικάνικη βιομηχανία παιχνιδιών να ανακάμψει μετά από την δεύτερη συντριβή της αγοράς το 1983

Η αρχή του 1980 είδε την "χρυσή" εποχή των παιχνιδιών για Arcade να φτάνει στο ζενίθ της. Οι πωλήσεις μηχανών arcade στην Αμερική αυξήθηκαν από τα 50 εκ. δολάρια το 1978 σε 900 εκ. το 1981, με κέρδος που να έχει φτάσει τα 2.8 δις. το 1980, 5 δις το 1981 και μέχρι τα 8 δις το 1982 , ξεπερνώντας το ετήσιο κέρδος της pop μουσικής(4 δις.), και των ταινιών Hollywood (3 δις) μαζί. Εκτός από αυτό, οι σπιτικές κονσόλες είχαν ένα κέρδος 4 δις την ίδια χρονιά. Αυτά τα κέρδη συνέχισαν μέχρι και το . Η μεγαλύτερη επιτυχία της εποχής ήταν το Pac-Man της εταιρίας Namco, το οποίο πώλησε πάνω από 350.000 μηχανές arcade. και μέσα σε έναν χρόνο έβγαλε έσοδα του 1 δις.

Η αρχή της δεκαετίας είδε την άνοδο του προσωπικού υπολογιστή και στην ανάπτυξη παιχνιδιών από ερασιτέχνες, ιδιαίτερα στην Ευρώπη με το ZX spectrum και στην Ασία με το NEC PC-88 και το MSX . Επίσης εμφανίστηκε η άνοδος της δημοσιογραφίας με θέμα τα βιντεοπαιχνίδια. Το 1983 ,ένας μεγάλος αριθμός εταιριών άρχισε να φτιάχνει όσο ποιο πολλά φθηνά παιχνίδια μπορεί, με στόχο να βγάζει μικρά έσοδα από το καθένα. Αυτό δημιούργησε έναν τεράστιο αριθμό shovelware τίτλων, με ποιό γνωστό αυτών το παιχνίδι E.T. για το Atari 2600 το οποίο ανατέθηκε σε έναν προγραμματιστή με χρονικό όριο 6 εβδομάδες και θεωρείτε σαν ένα από τα χειρότερα παιχνίδια όλων των εποχών, και έναν από τους κύριους λόγους για την πτώση της Atari. Αυτό κατέστρεψε την εμπιστοσύνη του κοινού στα βιντεοπαιχνίδια, και οδήγησε στην δεύτερη, και σημαντικότερη, συντριβή της αμερικανικής βιομηχανίας βιντεοπαιχνιδιών το 1984. Η βιομηχανία άρχισε να επανέρχεται χάρις την κυκλοφορία του Nintendo Entertainment System, το οποίο είχε σαν αποτέλεσμα η αγορά των κονσόλων για το σπίτι να κυριαρχείτε από Ιαπωνικές εταιρίες όπως η Nintendo και η Sega. Η Nintendo επέτρεπε

μονάχα σε συγκεκριμένες εταιρίες να φτιάχνουν παιχνίδια για το σύστημα της, και τους έδινε το Nintendo Seal of Approval, έτσι σταμάτησε κατά κύριο λόγο την δημιουργία των shovelware, και έτσι "έσωσε" την βιομηχανία από την συντριβή. Στο τέλος της δεκαετίας εμφανίστηκαν οι κονσόλες χειρός (handheld) Game Boy και Sega Game Gear, οι οποίες είχαν τεράστια επιτυχία. Το 1987 η Nintendo έχασε μία δίκη εναντίων της εταιρίας Blockbuster Entertainment, και αυτό επέτρεψε στα παιχνίδια να νοικιάζονται με τον ίδιο τρόπο όπως οι ταινίες.

1.2.3 1990

Εικόνα 1.4 : Το PlayStation ήταν η ποιο δημοφιλής κονσόλα της εποχής της, και έκανε δημοφιλή την χρήση CD για βιντεοπαιχνίδια.

Το 1990 εμφανίστηκαν πολλές καινοτομίες στην τεχνολογία των βιντεοπαιχνιδιών. Οι ποιό σημαντικές είναι

Η κατά κύριο λόγο χρήση CD στην διανομή

Την χρήση λογισμικού βασισμένου σε GUI, όπως το Amiga OS, Microsoft WIndows και Mac OS

Καινοτομίες στην τεχνολογία 3D γραφικών ,καθώς και την εστίαση στην ανάπτυξη παιχνιδιών με 3D γραφικά αντί για 2D

Η εξέλιξη σε ταχύτητα και τεχνολογία των CPU

Η σμίκρυνση του Hardware που οδήγησε στην άνοδο των βιντεοπαιχνιδιών για κινητά

Η άνοδος του internet που στο τέλος της δεκαετίας σήμανε την αρχή του online gaming

Αυτή την δεκαετία τα παιχνίδια βασισμένα σε ταινίες έγιναν πιο δημοφιλή, όπως και τα sequel βιντεοπαιχνιδιών. Επίσης, καθώς οι gamers της προηγούμενης δεκαετίας μεγάλωσαν, άρχισαν να εμφανίζονται παιχνίδια με ποιό ενήλικο και σοβαρό περιεχόμενο, με τα survival-horror games να έχουν μεγάλη ζήτηση. Με τις home consoles να κερδίζουν έδαφος, τα παιχνίδια Arcade είχαν ξεκινήσει την πτώση τους. Είδαν μια μικρή ελπίδα όταν τα fighting games Street Fighter 2 και

Mortal Kombat έκαναν μεγάλη επιτυχία, αλλά οι κονσόλες είχαν ήδη αρχίσει να φτάνουν ή ακόμα και να ξεπερνάνε τις μηχανές arcade σε δύναμη, και η τάση των παιχνιδιών να δέχονται 2-4 παίχτες, οδήγησε στον κόσμο να περιμένει να βγουν τα αντίστοιχα παιχνίδια στις κονσόλες, αντί να ξοδεύει κέρματα στα Arcade. Αυτό οδήγησε πολλά μαγαζιά με Arcade να κλείσουν.

Αυτή την δεκαετία ή Atari έκανε την τελευταία της προσπάθεια να μπει στον αγώνα τον κόνσολων με το Atari Jaguar, το οποίο δεν κατάφερε να κάνει επιθυμητές πωλήσεις, και η Atari άρχισε να ασχολείται μόνο με την ανάπτυξη παιχνιδιών, και όχι κόνσολων. Η Sega έβγαλε το Sega Saturn, αλλά εξαιτίας της πολύπλοκης αρχιτεκτονικής του, δεν είχε αρκετούς developer να φτιάχνουν παιχνίδια για αυτό, και έτσι χωρίς αρκετά μεγάλη βιβλιοθήκη τίτλων, δεν κατάφερε να κάνει καλές πωλήσεις. Η Nintendo έκανε συμφωνία με την Sony, να δημιουργήσει ένα CD player-add on για το Super NES, έτσι ώστε να μεγαλώσουν την διάρκεια ζωής της κονσόλας, αλλά τελικά η Nintendo ακύρωσε την συμφωνία και πήγε στην Panasonic για να τους φτιάξει το add on. Η Sony χρησιμοποίησε ότι έμαθε από την αρχιτεκτονική της κονσόλας της Nintendo και έφτιαξε την δικιά της κονσόλα, το Playstation. Το CD add on για το Super NES δεν βγήκε ποτέ, και η Nintendo δεν χρησιμοποίησε τεχνολογία CD για την επόμενη της κονσόλα, το Nintendo 64. Το αυξημένο κόστος και μικρή χωρητικότητα των cartridge εναντίων των CD έκανε το Nintendo 64 να χάσει έδαφος, και το Playstation να είναι η ποιό δημοφιλής κονσόλα αυτή την δεκαετία

Η βιομηχανία είχε ετήσια έσοδα των 20 δις. το 1993, και μέχρι το 1998 είχε φτάσει τα 30 δις.

1.2.4 2000

Εικόνα 1.5 : Η Sony συνέχισε να κερδίζει την αγορά με το PlayStation 2 το οποίο έγινε η top selling κονσόλα όλων των εποχών

Αυτή την δεκαετία η Sega έφυγε από την αγορά κόνσολων, η Nintendo έχασε ακόμα μεγαλύτερο μέρος της αγοράς, και η Microsoft έφτιαξε την δική της κονσόλα.

Η πρώτη κονσόλα αυτής της γενιάς ήταν το Sega Dreamcast, το οποίο παρόλο των καλών κριτικών που δέχτηκε, και των καινοτομιών που έκανε στα online παιχνίδια, απέτυχε εξαιτίας της κακής φήμης που είχε απόκτηση η Sega μετά την αποτυχία του Sega Saturn και της πειρατείας.

Η δεύτερη κονσόλα που βγήκε ήταν το Playstation 2, το οποίο εκτός από την εξέλιξη στον τομέα των γραφικών, ήταν η πρώτη κονσόλα που δεχόταν και ταινίες DVD, κάνοντας το έτσι ένα από τα φθηνότερα DVD players της εποχής του. Ένα χρόνο αργότερα η Nintendo έβγαλε το GameCube, την πρώτη τους κονσόλα που δέχεται οπτικούς δίσκους. Εξαιτίας της αργοπορίας του, το Ps2 είχε ήδη κερδίσει μεγάλο μερίδιο της αγοράς, και το GameCube δεν έκανε καλές πωλήσεις, και έτσι δεν είχε πολλές εταιρίες να φτιάχνουν παιχνίδια για αυτό, και έτσι δεν είχε μεγάλη επιτυχία. Τέλος η Microsoft μπήκε στην αγορά κόνσολων με το Xbox, το οποίο ήταν σχεδιασμένο για την ευκολία ανάπτυξης παιχνιδιών για το Xbox και το PC ταυτόχρονα.

Παρόλο που η βιομηχανία είχε παλιώσει, ήταν ακόμα πολύ ασταθείς, με εταιρίες ανάπτυξης βιντεοπαιχνιδιών να εμφανίζονται και να κλείνουν πολύ γρήγορα.

Παρόλα αυτά, στο τέλος της δεκαετίας ξεκίνησαν τα indie games να γίνονται ποιο δημοφιλή (με επιτυχίες όπως το Braid και το Limbo) . Τέλος έγινε δημοφιλής η ανάπτυξη παιχνιδιών για κινητά, τα οποία άρχισαν να αποκτούν δύναμη συγκρίσιμη με τα συστήματα χειρός όπως το Nintendo DS και το Sony PSP, και η ανάπτυξη βιντεοπαιχνιδιών για ιστοσελίδες κοινωνικών δικτύων, όπως η εταιρία Zynga που κατάφερε έσοδα πάνω από 300 εκ. μόνο δημιουργώντας παιχνίδια για το Facebook.

1.2.5 2010

Εικόνα 1.6 : Το 2013 ξεκίνησε η καινούργια γενιά κονσόλων, με το PS4 της Sony να σπάει το ρεκόρ πωλήσεων κονσόλας, με πάνω από 1 εκ. πωλήσεις 24 ώρες μετά το λανσάρισμα του.

Σήμερα, η βιομηχανία βιντεοπαιχνιδιών έχει μεγαλώσει σε μεγάλο βαθμό, διώχνοντας τελείως από πάνω της την ιδέα ότι τα βιντεοπαιχνίδια είναι μόνο για παιδιά. Με την εξέλιξη και την διαθεσιμότητα του internet, δόθηκε μεγάλη έμφαση στο Online Multiplayer και στην χρήση του Cloud για αποθήκευση δεδομένων όπως Save data, achievements κ.α.

Μετά την τεράστια επιτυχία του Nintendo Wii το 2007, εμφανίστηκε ένα μεγάλο ενδιαφέρον για χειρισμό παιχνιδιών με κίνηση (motion control). Με το motion controller της Microsoft για το Xbox 360, kinect, να έχει πάνω από 24 εκ. πωλήσεις η Microsoft αποφάσισε να το υποστηρίξει περισσότερο, δίνοντας το δωρεάν μαζί με την καινούργια τους κονσόλα, το Xbox One, μεγαλώνοντας έτσι το ενδιαφέρον των developers να φτιάξουν παιχνίδια για αυτό.

Ένας άλλος τομέας της τεχνολογίας την οποία άγγιξαν τα βιντεοπαιχνίδια ,είναι της εικονικής πραγματικότητας, με την επιτυχία της "προσωπίδας" Oculus Rift, και τις Valve και Sony να φτιάχνουν δικιές τους εκδοχές, έχει δημιουργηθεί ένας ενθουσιασμός στην αγορά για εμπειρίες που χρησιμοποιούν την εικονική

πραγματικότητα με ιδιαίτερο ενδιαφέρον στα horror games, στα οποία η απορρόφηση του παίχτη στο παιχνίδι είναι ένας από τους πιο σημαντικούς τομείς. Εκτός από την εικονική πραγματικότητα, παρατηρείτε ενδιαφέρον και στην Αυξημένη Πραγματικότητα (Augmented Reality), με παιχνίδια που χρησιμοποιούν τον πραγματικό κόσμο σαν το Setting του παιχνιδιού. Με όλο και περισσότερα augmented reality apps να δημιουργούνται για κινητά, παιχνίδια AR για το google glass ,και την Nintendo να υποστηρίζει απευθείας τα AR παιχνίδια, φτιάχνοντας AR κάρτες τις οποίες διαβάζει το 3DS, η Αυξημένη Πραγματικότητα είναι ένας τομέας με τον οποίο θα ασχοληθεί η βιομηχανία των βιντεοπαιχνιδιών στο μέλλον.
Αυτή την δεκαετία , τα παιχνίδια στο pc έχουν αρχίσει την άνοδο τους, μετά από το 2007 που θεωρήθηκαν "νεκρά", με την πλατφόρμα αγοράς ψηφιακών παιχνιδιών Steam να αποκτά δημοτικότητα χάρις τις συχνές και πολύ μεγάλες εκπτώσεις που δίνει στα ψηφιακά παιχνίδια, απέκτησε περισσότερους από 300 εκ. χρήστες τα τελευταία χρόνια, αναζωογονώντας την αγορά βιντεοπαιχνιδιών στα PC. Ταυτόχρονα σπρώχνοντας την αγορά προς τα ψηφιακά παιχνίδια, έδωσε ευκαιρία σε ανεξάρτητους developers να φτιάξουν ενδιαφέρον φθηνά παιχνίδια για χωρίς την βοήθεια εκδότη. Αυτά τα φθηνά μικρά βιντεοπαιχνίδια είχαν τόση πέραση στο κοινό, που στην E3 του 2013 η Sony έφερε ανεξάρτητους developers στην σκηνή να μιλήσουν και να προσκαλέσουν περισσότερους developers να φτιάχνουν ανεξάρτητα παιχνίδια και για τις κονσόλες. Με τις μεγάλες μηχανές γραφικών Unreal Engine, CryoEngine και Unity Pro να αλλάζουν τις τιμές τους από μια αγορά χιλιάδων ευρώ, σε μια συνδρομή μικρού κόστους κάθε μήνα, και την δημιουργία documentation και tutorial για αρχάριους, φαίνετε ότι τα ανεξάρτητα παιχνίδια θα συνεχίσουν την άνοδο στην υπόλοιπη δεκαετία.

1.3 Κατηγορίες παιχνιδιών

Στην αρχή της βιομηχανίας των παιχνιδιών, παιχνίδια τα οποία ήταν εμπνευσμένα ή χρησιμοποιούσαν παρόμοιο gameplay με άλλα, περιγραφόντουσαν σαν "κλώνοι" , με την λέξη Doom-Clone να περιγράφει όλα τα shooter παιχνίδια πρώτου προσώπου της εποχής, καθώς το παιχνίδι Doom ήταν το ποιο δημοφιλές της κατηγορίας. Νωρίς φάνηκε ότι αυτός ο τρόπος ονομασίας είναι προβληματικός και χρειάστηκε να δημιουργηθούν ευρύς κατηγορίες που να περιγράφουν καλύτερα τα παιχνίδια. Οι σημαντικότερες αυτών είναι :

1.3.1 Action

Εικόνα 1.7: Contra για το NES

Τα action games ζητάνε από τους παίχτες γρήγορα αντανακλαστικά, ευστοχία και συντονισμό για να ξεπεράσουν εμπόδια. Είναι η πιο βασική και ευρύ κατηγορία παιχνιδιών. Περιλάμβανε μεγάλες υποκατηγορίες όπως Shooter, Fighting, Sidescrollers , Platformers και πολλές άλλες. Το gameplay τους ως συνήθως βασίζετε σε μάχες

1.3.2 Adventure

Εικόνα 1.8: The Secret of Monkey Island για Pc

Τα Adventure Games ήταν κάποια από τα πρώτα παιχνίδια που δημιουργήθηκαν, με το Text Adventure παιχνίδι Colossal Cave Adventure να τους δίνει το όνομα τους το 1970. Τα adventure παιχνίδια βασίζονται στην ιστορία , διαλόγους και puzzle αντί για μάχες και περιμένουν τον παίχτη να σκεφτεί, ως συνήθως χωρίς χρονικό ώριο. Μετά το 2000 τα Adventure παιχνίδια άρχισαν να εξαφανίζονται, αλλά το 2010 είδαν μια αναγέννηση, με πολλές ευρωπαϊκές εταιρίες να ασχολούνται μαζί τους, και την εταιρία Double Fine, μια από τις ιστορικές εταιρίες adventure game να επιστρέφει σε αυτά

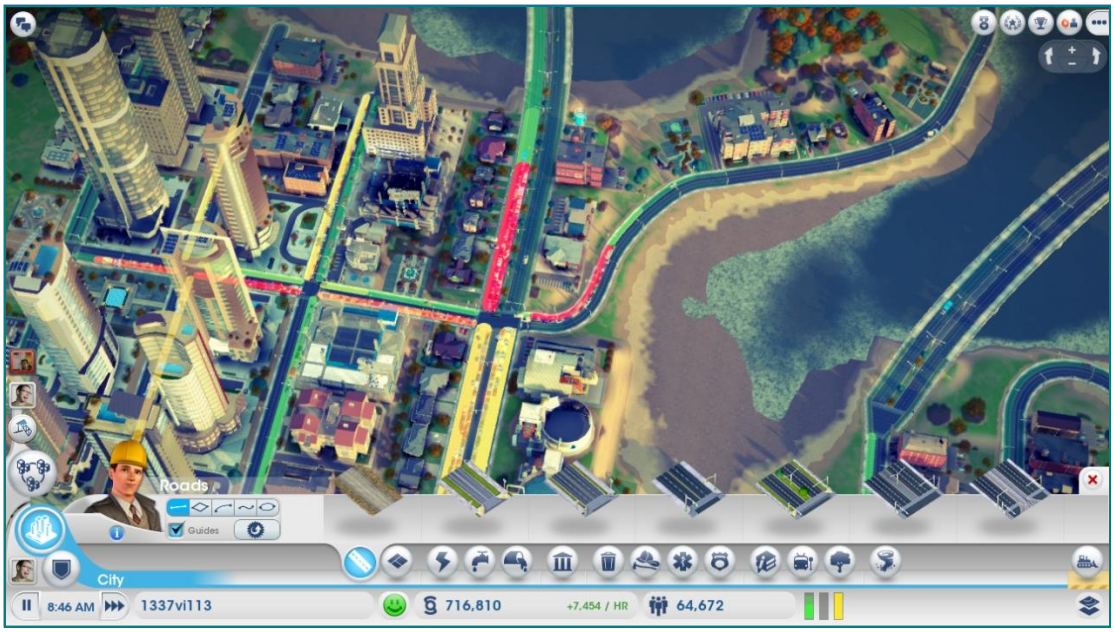
1.3.3 Role-Playing Games

Εικόνα 1.9 : Morrowind για PC

Τα roleplaying games βασίζουν το gameplay τους σε κλασικά επιτραπέζια παιχνίδια ρόλων όπως το Dungeons and Dragons. Στα περισσότερα rpg ο παίχτης διαλέγει ένα σετ ικανοτήτων , και καθώς παίζει το παιχνίδι κερδίζει πόντους

εμπειρίας , και χάρις αυτούς εκπαιδεύει περισσότερες ικανότητες. Η ιστορία μπορεί να είναι linear όπως στα περισσότερα ιαπωνικά rpg, ή μπορεί να δίνει στον παίχτη περισσότερη ελευθερία, δίνοντας του επιλογές για το πως θα αλλάξει την ιστορία Είναι μια από τις ποιο δημοφιλής κατηγορίες παιχνιδιών , και περιέχει άλλες μεγάλες υποκατηγορίες, όπως τα Jrpg (ιαπωνικά rpg) και τα mmorpg (massively multiplayer online rpg). Εκτός από αυτό , στοιχεία από τα rpg παιχνίδια έχουν αρχίσει να εμφανίζονται σε άλλες κατηγορίες, όπως συστήματα leveling σε shooter παιχνίδια.

1.3.4 Simulation Games



Εικόνα 1.10: SimCity 5 για το PC

Τα Simulation games είναι παιχνίδια που προσπαθούν να προσομοιώσουν πράγματα από την πραγματικότητα, και να βάλουν των παίχτη στην θέση του χειριστή. Τα Simulation games χρησιμοποιούνται συχνά στην εκπαίδευση εργαζομένων, όπως τα Flight Simulation games.

1.3.5 Strategy Games

Εικόνα 1.11 : Starcraft για το PC

Τα Strategy games εστιάζονται σε gameplay που χρειάζεται γρήγορη σκέψη και στρατηγική, και η κλίμακα τους μπορεί να είναι από μάχες με μικρές ομάδες, μέχρι την κατάκτηση του κόσμου. Ο παίχτης έχει τον έλεγχο του πεδίου της μάχης, και δίνει εντολές στις μονάδες του. Τα strategy games έχουν ρίζες από τα επιτραπέζια παιχνίδια στρατηγικής. Κυρίως τα παιχνίδια αυτής της κατηγορίας φτιάχνονται για PC, και περιλαμβάνουν υποκατηγορίες όπως turn based strategy στα οποία ο παίχτης έχει όσο χρόνο θέλει στην στροφή του, και real time strategy στα οποία παίζουν όλοι οι παίχτες ταυτόχρονα.

1.4 Τεχνολογικές εξελίξεις και το μέλλον

Εικόνα 1.12 : Το Oculus Rift

Η μεγαλύτερη τάση που βλέπουμε σήμερα είναι προς την εικονική πραγματικότητα, με VR visors όπως το Oculus Rift, και motion controls οι developers έχουν αρχίσει να φτιάχνουν εμπειρίες που έχουν τον παίχτη εντελώς αφοσιωμένο σε αυτές. Ένα από τα σημαντικότερα σημεία της εικονικής πραγματικότητας είναι ότι τα παιχνίδια χρειάζεται να παίζουν στα 60 fps σταθερά, αλλιώς ο παίχτης ζαλίζετε, άρα οι εξέλιξης της τεχνολογίας στις κάρτες γραφικών και CPU θα παίξουν σημαντικό ρόλο στην ανάπτυξη παιχνιδιών εικονικής πραγματικότητας. Τέλος οι τεχνολογίες motion control χρειάζεται να αυξήσουν την ακρίβεια τους, επειδή λάθη ακριβείας θα είναι πολύ πιο εμφανές σε Virtual Reality games.

Κεφάλαιο 2: Ανάπτυξη βιντεοπαιχνιδιών

Η ανάπτυξη εμπορικών βιντεοπαιχνιδιών ξεκίνησε το 1970 με την έλευση της πρώτης γενιάς κονσόλων και προσωπικών υπολογιστών. Λόγου των χαμηλών ικανοτήτων των υπολογιστών και κονσόλων, ένας προγραμματιστής μπορούσε να αναπτύξει ένα πλήρες παιχνίδι μόνος του. Ωστόσο με την εξέλιξη των ικανοτήτων των κονσόλων και υπολογιστών και την αύξηση των προσδοκιών των καταναλωτών, έγινε αναγκαία η ανάπτυξη βιντεοπαιχνιδιών από πολύ μεγαλύτερες ομάδες developers.

Τα βιντεοπαιχνίδια αναπτύσσονται γενικά σε φάσεις. Πρώτον στην προ-παραγωγή (pre-production) , pitches πρωτότυπα και game design documents δημιουργούνται. Αν το παιχνίδι εγκριθεί, ξεκινάει η φάση ανάπτυξης. Σε αυτή την φάση απασχολούνται μεγάλες ομάδες developer , διαφόρων αρμοδιοτήτων, όπως σχεδιαστές, καλλιτέχνες, προγραμματιστές ,testers κλπ. Τα παιχνίδια περνάνε από τα στάδια ανάπτυξης,alpha και beta πριν διαφημιστούν με demo και trailers, και τελικά εκδοθούν.

2.1 Ρόλοι Ανάπτυξης

Στις αρχές της βιομηχανίας των βιντεοπαιχνιδιών, ένας προγραμματιστής έπαιρνε σχεδόν όλους τους ρόλους της ανάπτυξης ενός παιχνιδιού, από τον προγραμματισμό, γραφικά, ηχητικά εφέ κλπ, και ο χρόνος ανάπτυξης ενός παιχνιδιού ήταν το πολύ μερικοί μήνες . Σήμερα οι ομάδες ανάπτυξης έχουν εξειδικευμένους developer για κάθε ρόλο στην ανάπτυξη του παιχνιδιού. Μερικοί από τους κύριους ρόλους είναι :

2.1.1 Παραγωγός

Ο παραγωγός διαχειρίζεται την ομάδα ανάπτυξης, τα χρονοδιαγράμματα, τα reports προόδου, τις προσλήψεις και ανάθεση προσωπικού μεταξύ άλλων υποχρεώσεων. Είναι αυτός που επιβλέπει αν η ομάδα βρίσκετε εντός σχεδίου, προϋπολογισμού, και αν δεν υπάρχουν προβλήματα μέσα στην ομάδα.

2.1.2 Σχεδιαστής

Ο σχεδιαστής είναι το άτομο το οποίο σχεδιάζει το gameplay, τους κανόνες και την δομή του παιχνιδιού. Οι ομάδες ανάπτυξης ως συνήθως έχουν έναν επικεφαλή σχεδιαστή, ο οποίος συντονίζει το έργο των άλλων σχεδιαστών. Είναι αυτός που οραματίζετε πως θα "παίζει" το παιχνίδι. Συχνά ο σχεδιαστής παίρνει και τον ρόλο του συγγραφέα, ή δουλεύει μαζί με αυτόν, για να δημιουργήσουν την ιστορία, διαλόγους και αφήγηση του παιχνιδιού. Σε μεγαλύτερες ομάδες, υπάρχουν ξεχωριστοί σχεδιαστές για διάφορα μέρη του παιχνιδιού, όπως gameplay, UI, διάλογοι κ.α.

2.1.3 Καλλιτέχνης

Ο καλλιτέχνης δημιουργεί artwork για το παιχνίδι. Η παραγωγή του artwork συνήθως επιβλέπετε από έναν επικεφαλή καλλιτέχνη ο οποίος σιγουρεύει ότι το οι καλλιτέχνες δεν ξεφεύγουν από το όραμα που έχει σχεδιαστεί για το παιχνίδι. Η δουλεία του καλλιτέχνη μπορεί να είναι βασισμένη σε 2D ή 3D. Οι καλλιτέχνες 2D παράγουν εικώνες(sprites), υφές(textures), φόντα , UI (user interface) κ.α. Οι καλλιτέχνες 3D παράγουν 3D μοντέλα , meshes, animations, cinematics κ.α. Μερικοί καλλιτέχνες καταλαμβάνουν και τους δύο ρόλους.

2.1.4 Προγραμματιστής

Ο προγραμματιστής είναι αυτός που γράφει τον κώδικα του παιχνιδιού, ή της game engine που θα χρησιμοποιηθεί. Υπάρχει συνήθως ένας ή και περισσότεροι επικεφαλείς προγραμματιστές οι οποίοι ξεκινάνε την δημιουργία του κώδικα του παιχνιδιού και επιβλέπουν την δουλειά των υπόλοιπων προγραμματιστών. Εξειδικευμένοι ρόλοι προγραμματιστών μέσα στο έργο περιλαμβάνουν

-Φυσική : ο προγραμματισμός του game engine , που μέσα της εμπεριέχετε η μηχανή φυσικής, κίνηση αντικειμένων, εντοπισμός συγκρούσεων κ.α.

-AI: Φτιάχνουν την συμπεριφορά των αντικειμένων μέσα στο παιχνίδι, γράφοντας scripts για αυτά.

-Γραφικά: Η διαχείριση γραφικού περιεχομένου και σωστή χρήση της μνήμης, δημιουργία μηχανής γραφικών που θα δέχεται και διαχειρίζεται τα κομμάτια (asset) του παιχνιδιού

-Ήχος: Ενσωμάτωση της μουσικής, των διαλόγων και ηχητικών εφέ στις σωστές θέσεις και χρόνους.

-Gameplay: Εφαρμογή των διάφορων κανόνων και χαρακτηριστικών του παιχνιδιού

-Scripting: Ανάπτυξη και συντήρηση ενός συστήματος εντολών υψηλού επιπέδου για διάφορες χρήσης μέσα στο παιχνίδι, όπως το AI

-UI: παραγωγή στοιχείων διεπαφής χρήστη όπως μενού επιλογών , HUDS, μενού βοήθειας κ.α.

- Επικοινωνία Δικτύου: η διαχείριση της εισόδου και εξόδου δεδομένων για τοπικό ή διαδικτυακό gameplay

-Game Tools: η παραγωγή εργαλείων τα οποία βοηθούν στην ανάπτυξη του παιχνιδιού, ιδιαίτερα για τους σχεδιαστές και τους scripters.

2.1.5 Σχεδιαστής Επιπέδων

Ο σχεδιαστής επιπέδων χρησιμοποιεί διάφορα προγράμματα (που πιθανός έχουν φτιαχτεί από τους προγραμματιστές για το συγκεκριμένο έργο) και το game engine για να φτιάξει επίπεδα για το παιχνίδι. Οι προγραμματιστές συνήθως σχεδιάζουν εργαλεία υψηλού επιπέδου για την σχεδίαση επιπέδων, έτσι ώστε οι σχεδιαστές να μην χρειάζεται να διαχειρίζονται τον κώδικα του παιχνιδιού. Πολλές φορές οι σχεδιαστές επιπέδων χρησιμοποιούν γλώσσες scripting υψηλού επιπέδου για διαδραστικά περιβάλλοντα ή AI. Σε αντίθεση με τα εργαλεία δημιουργίας επιπέδων που συχνά παρέχονται στην κοινότητα, οι σχεδιαστές συχνά δουλεύουν με placeholders και πρωτότυπα πριν ετοιμαστούν τα art assets

2.1.6 Ηχολήπτης

Οι ηχολήπτες είναι υπεύθυνοι για τα ηχητικά εφέ και την τοποθέτηση τους στον χώρο. Πολλές φορές επιβλέπουν το voice acting και την δημιουργία άλλων ηχητικών εφέ. Συνθέτες που δημιουργούν την μουσική επένδυση του παιχνιδιού περιλαμβάνονται στην ομάδα ήχου, αν και συχνά αυτό το έργο ανατίθεται σε άλλες εταιρίες, έξω από την ομάδα ανάπτυξης

2.1.7 Δοκιμαστής

Η διασφάλιση της ποιότητας (quality assurance) πραγματοποιείτε από τους δοκιμαστές του παιχνιδιού. Ένας δοκιμαστής αναλύει το παιχνίδι για να τεκμηριώσει τα ελαττώματα του ως τμήμα τού ποιοτικού ελέγχου. Η δοκιμή είναι ένας εξαιρετικά τεχνικός τομέας που απαιτεί εμπειρία στην χρήση υπολογιστών, και αναλυτική επάρκεια. Οι δοκιμαστές εξασφαλίζουν ότι το παιχνίδι εμπίπτει εντός του προτεινόμενου σχεδίου , και από την άποψη ότι λειτουργεί σωστά, αλλά και ότι είναι διασκεδαστικό. Αυτό περιλαμβάνει τον έλεγχο όλων των δυνατοτήτων , τη συμβατότητα, το localization κ.α. Παρά το γεγονός ότι η δοκιμή είναι αναγκαία καθόλα την διάρκεια της ανάπτυξης, είναι ακριβός και άρα πολλές φορές γίνετε μόνο κατά το τέλος της.

2.2 Φάσεις Ανάπτυξης

2.2.1 Προ-παραγωγή

Η προ-παραγωγή είναι μια φάση σχεδιασμού του έργου που επικεντρώνεται στην ιδέα και την ανάπτυξη ιδεών και την παραγωγή αρχικών εγγράφων σχεδιασμού. Ο στόχος της ανάπτυξης της βασικής ιδέας είναι να παράγει σαφή και εύκολη κατανοητή τεκμηρίωση η οποία περιγράφει τις εργασίες, χρονοδιαγράμματα και εκτιμήσεις για την ομάδα αναπτύξεις. Μπορεί να χωριστεί σε 3 στάδια:

High Concept

Είναι η περιγραφή του παιχνιδιού με μερικές προτάσεις

Pitch

Μια σύντομη έγγραφη περίληψη που παρουσιάζει τα selling points του παιχνιδιού και με λεπτομέρεια εξηγεί γιατί η ανάπτυξη του παιχνιδιού θα είναι επικερδής

Concept

Είναι ένα πιο λεπτομερές έγγραφο από το pitch. Αυτό περιλαμβάνει όλες τις πληροφορίες για το παιχνίδι, όπως το High Concept, περιγραφή του gameplay, τα χαρακτηριστικά, την ιστορία, το κοινό το οποίο θα στοχεύσει, πλατφόρμες hardware, το εκτιμώμενο χρονοδιάγραμμα, την ανάλυση του εμπορίου, τις απαιτήσεις της ομάδας και την ανάλυση κινδύνου

Έγγραφο σχεδιασμού του παιχνιδιού

Πριν μπορέσει να ξεκινήσει η παραγωγή, η ομάδα ανάπτυξης παράγει την πρώτη έκδοση ενός εγγράφου σχεδιασμού το οποίο περιλαμβάνει όλο ή το μεγαλύτερο μέρος του pitch . Το έγγραφο σχεδιασμού περιγράφει το concept του παιχνιδιού, και όλο το gameplay λεπτομερώς. Μπορεί επίσης να περιλαμβάνει προσχέδια διαφόρων πτυχών του παιχνιδιού. Το έγγραφο σχεδιασμού συνοδεύεται μερικές φορές από πρωτότυπα κάποιων κομματιών του παιχνιδιού για να πάρει η ομάδα μια καλύτερη ιδέα για το πως πρέπει να λειτούργει το παιχνίδι. Το έγγραφο σχεδιασμού παραμένει ένα "ζωντανό" έγγραφο και αλλάζει συχνά κατά την διάρκεια της ανάπτυξης

2.2.2 Παραγωγή

Η παραγωγή είναι το βασικό στάδιο της ανάπτυξης, στο οποίο δημιουργούνται τα asset και ο κώδικας του παιχνιδιού. Κατά την διάρκεια της πραγματοποιούνται οι εξής διαδικασίες:

Σχεδιασμός

Ο σχεδιασμός του παιχνιδιού είναι μια ουσιαστική και συνεργατική διαδικασία στην οποία σχεδιάζονται τα περιεχόμενα και οι κανόνες του παιχνιδιού. Απαιτεί καλλιτεχνική και τεχνική επάρκεια καθώς και δεξιότητα στην γραφή. Κατά την διάρκεια της ανάπτυξης οι σχεδιαστές εφαρμόζουν και τροποποιούν τον σχεδιασμό του παιχνιδιού, έτσι ώστε να αντικατοπτρίζει το σημερινό όραμα του. Χαρακτηριστικά και επίπεδα συχνά αφαιρούνται ή προστίθενται. Μπορεί να αλλάξει το art style, η ιστορία, ή να προστεθεί μια νέα πλατφόρμα. Για αυτές τις αλλαγές πρέπει να ενημερώνετε το έγγραφο σχεδιασμού.

Προγραμματισμός

Ο προγραμματισμός του παιχνιδιού γίνεται από έναν η περισσότερους προγραμματιστές. Αυτοί αναπτύσσουν πρωτότυπα για να δοκιμάσουν ιδέες, πολλές από τις οποίες μπορεί να μην μπουν στο τελικό παιχνίδι. Οι προγραμματιστές ενσωματώνουν νέα χαρακτηριστικά που απαιτούνται από τον σχεδιασμό του παιχνιδιού και διορθώνουν τυχών σφάλματα που εισάγονται κατά την ανάπτυξη. Ακόμα και αν χρησιμοποιείτε έτοιμο game engine "off-the-shelf" , χρειάζεται ένα μεγάλο μέρος προγραμματισμού για να προσαρμοστεί για το παιχνίδι που αναπτύσσετε

Δημιουργία επιπέδων

Από άποψη χρόνου, το πρώτο επίπεδο παίρνει τον περισσότερο χρόνο για να αναπτυχθεί. Κατά την διάρκεια της ανάπτυξης οι σχεδιαστές και καλλιτέχνες ζητάνε την δημιουργία η τροποποίηση εργαλείων για την διευκόλυνση δημιουργίας επιπέδων, τα οποία επιτρέπουν ταχύτερη και υψηλότερη ποιότητα ανάπτυξης. Νεοεισαχθέντα χαρακτηριστικά του παιχνιδιού μπορεί να αναγκάσουν επίπεδα που σχεδιάστηκαν πριν την αλλαγή να επανασχεδιαστούν ή βγουν τελείως από το παιχνίδι.

Παραγωγή Σχεδίων

Οι καλλιτέχνες ξεκινάνε να φτιάχνουν τα assets σύμφωνα με το concept art που φτιάχτηκε στην φάση της σχεδίασης. Κατά την διάρκεια της παραγωγής, τα assets που φτιάχνουν αντικαθιστούν τα υπάρχοντα place holders που υπάρχουν.

Παραγωγή ήχου

Ο ήχος του παιχνιδιού μπορεί να χωριστεί σε τρείς κατηγορίες : ηχητικά εφέ, μουσική και voice over. Η παραγωγή ηχητικών εφέ γίνετε είτε με μικροαλλαγές σε ένα δείγμα ήχου, η στην αναπαραγωγή του ήχου με πραγματικά αντικείμεναΣε μεγάλα παιχνίδια με πολλούς διαλόγους, το κόστος του voice acting μεγαλώνει υπερβολικά, για αυτό συχνά μόνο συγκεκριμένοι σημαντικοί χαρακτήρες έχουν φωνές. Η μουσική μπορεί να συντίθεται με υπολογιστές, ή να ηχογραφείτε ζωντανά.

Δοκιμές

Η διασφάλιση της ποιότητας διαδραματίζει σημαντικό ρόλο στην ανάπτυξη παιχνιδιών. Οι δοκιμαστές αρχίζουν να δουλεύουν όταν οποιοδήποτε κομμάτι του παιχνιδιού να λειτουργεί. Αυτό μπορεί να είναι ένα επίπεδο, ή ένα υποσύνολο του λογισμικού που μπορεί να χρησιμοποιηθεί σε οποιοδήποτε βαθμό (π.χ. το βασικό σύστημα μάχης ενός παιχνιδιού μπορεί να είναι έτοιμο πριν ετοιμαστεί κάποιο επίπεδο, και χρειάζεται δοκιμή). Όσο πλησιάζει το τέλος της ανάπτυξης, η ομάδα δοκιμών μεγαλώνει για να ανταποκριθεί στο όλο και μεγαλύτερο φόρτο εργασίας που δημιουργείτε

2.2.3 Milestones

Μπορεί να απαιτηθεί από εμπορικά έργα ανάπτυξης παιχνιδιών να καλύψουν ενδιάμεσους στόχους που έχουν καθοριστεί από τον εκδότη. Τα Milestones σηματοδοτούν σημαντικά γεγονότα κατά την διάρκεια ανάπτυξης του παιχνιδιού και χρησιμοποιούνται για την παρακολούθηση της προόδου του. Τέτοια ορόσημα μπορεί να είναι για παράδειγμα τα first playable,alpha και beta εκδώσεις του παιχνιδιού. Δεν υπάρχει πρότυπο στην βιομηχανία για τον καθορισμού των milestones, και άρα αλλάζουν από εκδότη σε εκδότη, αλλά αυτά είναι μερικά milestones που χρησιμοποιούνται συχνά.:

First Playable (pre-Alpha)

Αυτή είναι μια εκδοχή του παιχνιδιού η οποία έχει αντιπροσωπευτικά gameplay και assets, αυτή είναι η πρώτη έκδοση στην οποία κύρια στοιχεία του gameplay

λειτουργούν. Είναι συχνά βασισμένη σε κάποιο πρωτότυπο που δημιουργήθηκε στην προ-παραγωγή.

Alpha

Αυτό είναι το στάδιο στο οποίο το βασικό gameplay,τα κύρια χαρακτηριστικά του παιχνιδιού και τα περισσότερα art assets έχουν τελειώσει. Το παιχνίδι είναι "Feature Complete" όσον αφορά τα κύρια χαρακτηριστικά που αποφασίστηκαν στην σχεδίαση, αλλά μπορούν ακόμα να προστεθούν μικρότερα νέα χαρακτηριστικά.

Code freeze

Σε αυτό το στάδιο δεν προστίθεται καινούργιος κώδικας στο παιχνίδι, αλλά μονάχα διορθώνονται game breaking σφάλματα.

Beta

Το Beta είναι το "Feature Complete" και "Asset Complete" στάδιο του παιχνιδιού και το μόνο που μένει είναι να διορθωθούν τα σφάλματα. Αυτή η εκδοχή δεν περιέχει σφάλματα που θα έκαναν το παιχνίδι να κριθεί μη πολύσημο. Δεν γίνονται άλλες αλλαγές στα χαρακτηριστικά του παιχνιδιού, τα assets ή στον κώδικα

Code Release

Σε αυτό το στάδιο όλα τα σφάλματα έχουν φτιαχτεί και το παιχνίδι είναι έτοιμο να ξεκινήσει παραγωγή ή να γίνει ο τελευταίος έλεγχος σε σχέση με το σχέδιο δοκιμής QA

Gold Master

Το Gold Master είναι το τελευταίο στάδιο του παιχνιδιού και χρησιμοποιείται ως υπόδειγμα για την παραγωγή του παιχνιδιού

Μετά την παραγωγή (Post-Production)

2.2.4 Συντήρηση

Μόλις ένα παιχνίδι εκδοθεί, η φάση συντήρησης για το παιχνίδι αρχίζει. Οι προγραμματιστές περιμένουν ένα χρονικό διάστημα για να πάρουν όσες περισσότερες αναφορές σφαλμάτων μπορούν. Μόλις μαζευτούν αρκετές αναφορές σφαλμάτων, ή αν εμφανιστεί ένα πολύ σοβαρό σφάλμα αρχίζουν να δουλεύουν στο patch. Το patch μπορεί να χρειαστεί εβδομάδες ή μήνες για να αναπτυχθεί, αλλά χρειάζεται για να διορθώσει τα προβλήματα που ξέφυγαν από το QA testing.

2.3 Game Engines

2.3.1 Εισαγωγή στην ανάπτυξη του παιχνιδιού με game engine

Τα game engines είναι πλαίσια λογισμικού, τα οποία είναι σχεδιασμένα για την ανάπτυξη βιντεοπαιχνιδιών. Ως συνήθως αποτελούνται από μια μηχανή rendering 3D ή 2D γραφικών, μηχανή φυσικής, μηχανή έλεγχου ολισθήσεων και λειτουργίες οι οποίες διευκολύνουν τους developer στην πρόσθεση ήχου, scripts, animations, AI, διαχείριση μνήμης, porting σε άλλες πλατφόρμες και ότι άλλο βοηθήσει στην ανάπτυξη του συγκεκριμένου τίτλου.

Καθώς τα διαφορετικά είδη βιντεοπαιχνιδιών έχουν τόσο διαφορετικό gameplay, οι developers βιντεοπαιχνιδιών έχουν ανάγκη από διαφορετικά "εργαλεία" ανάλογα το είδος του παιχνιδιού που φτιάχνουν. Για αυτό τον λόγο οι μεγάλες εταιρίες δημιουργούν τις δικές τους game engines, που θα τους δίνουν ακριβώς ότι χρειάζεται το βιντεοπαιχνίδι που φτιάχνουν. Για τους μικρότερους developers που δεν έχουν την πείρα και τον χρόνο να φτιάξουν την δική τους game engine, υπάρχουν έτοιμες game engines που μπορούν να χρησιμοποιήσουν, οι οποίες ως συνήθως εξειδικεύονται στην ανάπτυξη κάποιου είδους παιχνιδιού

Μέχρι τώρα οι game engines στόχευαν να πωληθούν σε εταιρίες, και για αυτό οι τιμές τους ήταν μεγαλύτερες από ότι μπορούσε να πληρώσει κάποιος ερασιτέχνης game developer, αλλά με την στροφή που έχει πάρει η αγορά τα τελευταία χρόνια προς τα indie games, τα game engines έχουν αρχίσει να δίνουν ποιό λογικές τιμές για ιδιώτες, όπως μια μικρή μηνιαία συνδρομή, η ένα free version της μηχανής

2.3.2 Διαθέσιμες μηχανές και σύγκριση μεταξύ τους

Καθώς το κάθε είδος παιχνιδιού έχει τόσο διαφορετικές ανάγκες, έχουν δημιουργηθεί μηχανές οι οποίες εξειδικεύονται στην ανάπτυξη ενός είδος παιχνιδιού, π.χ. η SCUMM engine που χρησιμοποιήθηκε σε adventure παιχνίδια από την Lucas Arts το 1987

Εικόνα 2.1 : Indiana Jones and the Fate of Atlantis, ένα παιχνίδι δημιουργημένο με την SCUMM engine

Πέρα από το είδος των παιχνιδιών που υποστηρίζουν, οι game engine χωρίζονται στην ελευθερία προς ευκολία χρήσης που δίνουν στον παίχτη. Για παράδειγμα μηχανές όπως οι Game Maker και Construct 2 είναι πολύ εύκολες στην χρήση και δεν προαπαιτούν γνώσεις προγραμματισμού , αλλά δίνουν μικρή ελευθερία στις επιλογές πού έχει ο developer.
Μηχανές όπως οι Unity, CryEngine και Unreal Engine, είναι φτιαγμένες σαν ποιό γενικές μηχανές με τις οποίες μπορούν να φτιαχτούν πολλά διαφορετικά είδη παιχνιδιών, και σου δίνουν αρκετή ελευθερία , αλλά ζητάνε τουλάχιστον βασικές γνώσεις προγραμματισμού. Αυτές οι μηχανές προτιμούνται σε action και ιδιαίτερα First Person Shooter παιχνίδια, αλλά δίνουν αρκετή ελευθερία ώστε να μπορούν να αναπτυχτούν και άλλα είδη παιχνιδιών χωρίς ιδιαίτερη δυσκολία.

2.3.3 Ανάπτυξη 2D και 3D παιχνιδιών με τη μηχανή unity

Εμείς για την ανάπτυξη του βιντεοπαιχνιδιού, θα χρησιμοποιήσουμε την μηχανή Unity. Η Unity είναι μια μηχανή που επιτρέπει ανάπτυξη βιντεοπαιχνιδιών σε πολλές πλατφόρμες(cross-platform), με ενσωματωμένο IDE, η οποία δημιουργήθηκε από την Unity Technologies.

Εικόνα 2.2 : Το πολυβραβευμένο παιχνίδι Gone Home φτιάχτηκε με την μηχανή Unity

Μηχανή Γραφικών

Η μηχανή γραφικών της Unity χρησιμοποίει Direct3D(για windows και Xbox 360), OpenGL(για Mac,Windows και Linux), και OpenGL ES(για android και iOS). Υποστηρίζει τεχνικές για την δημιουργία 3D γραφικών όπως bump mapping, reflection mapping, parallax mapping, δυναμικές σκιές και άλλες. Η unity υποστηρίζει assets και είδη αρχείων από τα προγράμματα : 3ds Max, Maya, Softimage,Blender, ZBrush, Adobe Photosphop, Adobe Fireworks και άλλα. Τα assets αυτά μπορούν να προστεθούν στο project και να τις χειριστεί ο developer μέσα από το UI της unity. Από το Version 4.2, η Unity υποστηρίζει και 2D γραφικά, με υποστηρίξει εικόνων sprite και καλύτερη υποστήριξη για animation εικόνων μέσο της τεχνολογίας Mecanim.

Εγγραφή Σεναρίων (Scripting)

Το scripting της unity είναι φτιαγμένο στο Mono. Οι προγραμματιστές μπορούν να χρησιμοποιήσουν JavaScript, C# ή Boo. Με το version 3 η unity περιλαμβάνει ένα custom version του MonoDevelop για το γράψιμο των script.

Cross-Platform

Εικόνα 2.3 : Το παιχνίδι Pid φτιάχτηκε με την Unity, για τις πλατφόρμες OS X, PlayStation 3, Windows & Xbox 360.

Η unity υποστηρίζει ταυτόχρονη ανάπτυξη σε πολλαπλές πλατφόρμες. Επίσης υποστηρίζει την επιλογή συμπίεσης texture, και ανάλυσης, για κάθε πλατφόρμα ξεχωριστά. Αυτή την στιγμή υποστηρίζει τις πλατφόρμες : BlackBerry 10, Windows 8, Windows Phone 8, Windows, Mac, Linux, Android,iOS, Unity Web Player, Adobe Flash, PlayStation 3, Xbox 360, Wii U Wii και PlayStation Vita. Μελλοντικά θα υποστηρίζει PlayStation 4 και Xbox One

Μηχανή Φυσικής

Η unity έχει ενσωματωμένη υποστήριξη για την μηχανή φυσικής PhysX της nvidia. Με το version 4.2, προστέθηκε στην Unity υποστήριξη για την μηχανή φυσικής 2D παιχνιδιών, Box2D.

Η unity προτιμάτε από τους αρχάριους game developer και μαθητές για την ευκολία χρήσης της σχετικά με τις δυνατότητές της, για την ύπαρξη free version της, τον μεγάλο όγκο documentation που υπάρχει για αυτήν και για την δυνατότητα ανάπτυξης για πολλές πλατφόρμες με ευκολία.

Κεφάλαιο 3: Παρουσίασή των βασικών λειτουργιών της μηχανής unity

3.1 Σχεδίαση του παιχνιδιού

Το παιχνίδι που θα φτιάξουμε θα είναι ένα 2D Platformer με 3 επίπεδα, έναν boss και ένα επίπεδο που ενώνει τα υπόλοιπα (hub level) με το οποίο ο παίχτης θα μπορεί να επιλέξει σε ποιό επίπεδο θέλει να πάει. Ο παίχτης θα μπορεί να πηδάει, και θα έχει την ικανότητα να πηδήξει μια ακόμα φορά όσο βρίσκετε στον αέρα, θα έχει μια κοντινή επίθεση με το σπαθί του, και θα μπορεί να πετάει φλόγες από το χέρι του για μακρινή επίθεση, η οποία θα έχει κάποιο χρόνο επαναφόρτισης. Θα μπορεί να βρίσκει "Power Ups" στα επίπεδα τα οποία θα αλλάζουν την μακρινή του επίθεση., και στο HUD θα φαίνετε το πόση ζωή του απομένει.

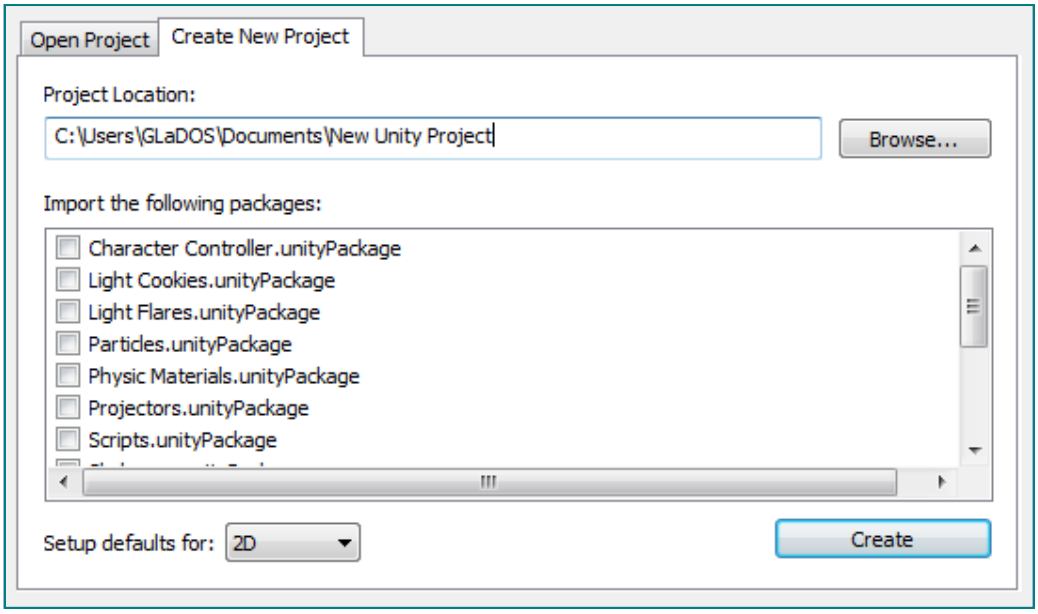
Το κάθε επίπεδο θα παρουσιάζει και κάποιο καινούργιο κομμάτι gameplay, όπως διακόπτες , κινούμενες πλατφόρμες ή την αλλαγή της κάμερας. Θα υπάρχει μια γκάμα αντιπάλων με διαφορετικές συμπεριφορές, όπως αντίπαλοι που θα κινούνται δεξιά και αριστερά στις πλατφόρμες, ή θα κυνηγάνε τον παίχτη. Θα υπάρχει ένα σύστημα "δέντρου" διαλόγων, στους οποίους ο παίχτης θα μπορεί να επιλέξει την απάντηση του. Η λογική του boss θα είναι να χτυπάει τον παίχτη με μακρινές επιθέσεις, και να αλλάζει τοποθεσία κάθε φορά που τον χτυπάει ο παίχτης. Η ζωή του boss θα εμφανίζετε στην οθόνη όταν ξεκινάει η μάχη.

Παρακάτω θα δείξουμε πως μπορούμε να φτιάξουμε την βασική κίνηση του παίχτη με χρήση της unity και scripts, καθώς και άλλες χρήσιμες λειτουργίες της που θα μας χρειαστούν για την δημιουργία του παιχνιδιού, όπως το πως μπορούμε να χρησιμοποιήσουμε τον εντοπισμό συγκρούσεων

3.2 Το περιβάλλον ανάπτυξης

3.2.1 Παρουσίασή του Editor και του workspace

Μόλις ξεκινάμε ένα νέο project στην Unity, εμφανίζετε το Project Wizard από το οποίο μπορούμε να διαλέξουμε τι πακέτα με asset να κάνει import στο project μας.



Εικόνα 3.1 : Project Wizard

Για το παιχνίδι που φτιάχνουμε θα χρησιμοποιήσουμε δικά μας assets, αρα δεν επιλέγουμε κανένα. Το setup defaults κάνει διάφορα settings να είναι πιο βολικά για ανάπτυξη 2D ή 3D παιχνίδια (π.χ. κάνει τις εικόνες import σαν sprites κατευθείαν). Επιλέγουμε setup defaults 2D και Create

Όταν φορτώσει το νέο project παρατηρούμε διάφορα tabs. Πρέπει να έχουμε τα:

Hierarchy: Μια λίστα με τα GameObjects που βρίσκονται στο scene

Scene: Εδώ μπορούμε να δούμε τα GameObjects στον τρισδιάστατο χώρο. Παρόλο που εργαζόμαστε στις διαστάσεις x και y, η διάσταση z δεν παύει να υπάρχει και μπορούμε να την χρησιμοποιήσουμε. Παιχνίδια σαν αυτό είναι γνωστά σαν 2.5D ή pseudo-3D

Game:Αυτό το Tab δείχνει το scene από το Point of View της main camera. Εδώ βλέπουμε τι θα βλέπει ο παίχτης όταν τρέχει το παιχνίδι, και θα μπορούμε από

εδώ να ελέγξουμε τον παίχτη, όταν γράψουμε το Script για την κίνηση του. Όταν πατάμε το κουμπί Play, το Game tab επιλέγετε αυτόματα

Inspector:Επιλέγοντας ένα GameObject (π.χ. την Main Camera) , μπορούμε να δούμε στο Inspector τις ιδιότητες του κάθε component του και να τις αλλάξουμε. Μπορούμε να αλλάξουμε αυτές τις ιδιότητες και ενώ παίζει το παιχνίδι για να τεστάρουμε διάφορες τιμές, αλλά όταν πατάμε το stop, οι αλλαγές που κάναμε κατά την διάρκεια του play αναιρούνται.

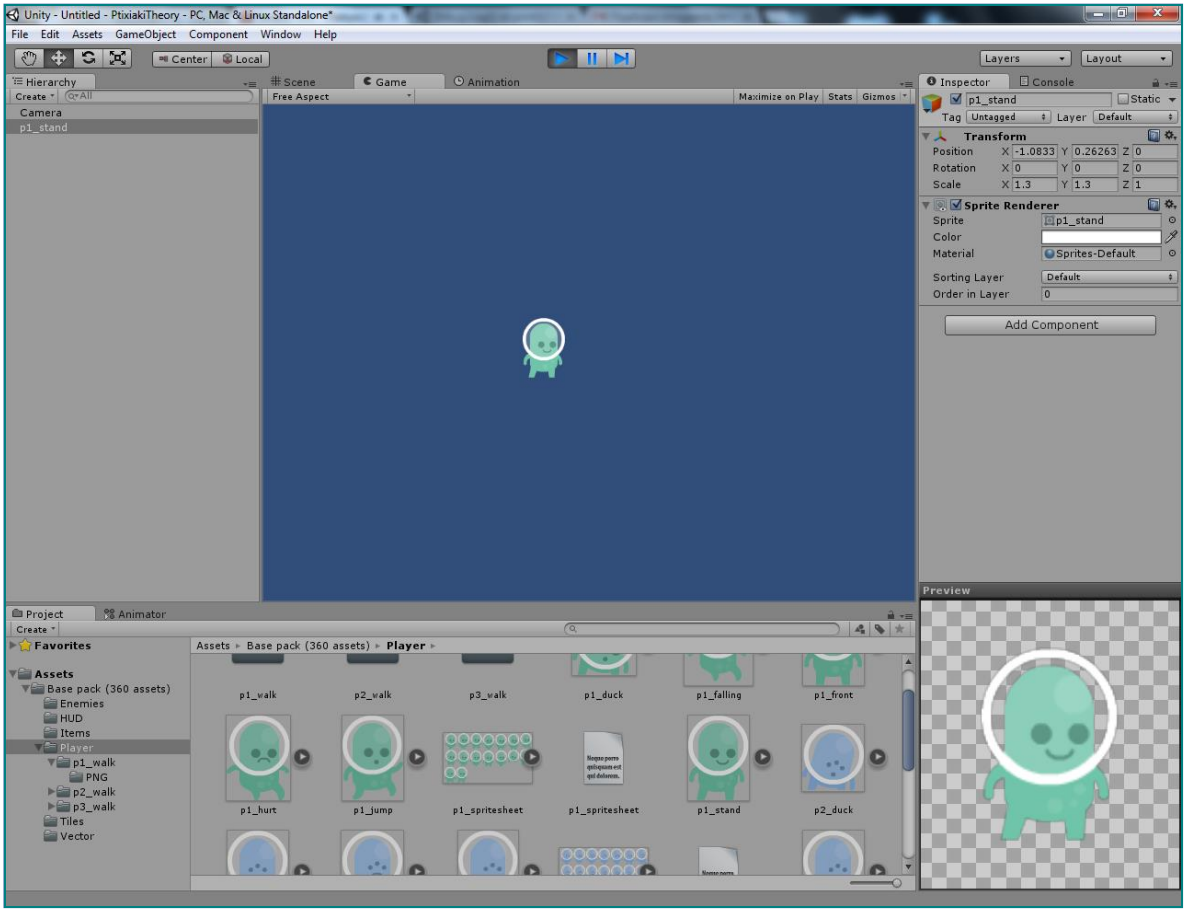
Console:Εδώ θα βλέπουμε τα μηνύματα που θα πετάνε τα scripts

Project: Σε αυτό το tab βλέπουμε τους φακέλους με τα περιεχόμενα του φακέλου Assets του project. Το path αυτού του φακέλου το διαλέξαμε στον project wizard.

Εκτός από αυτά τα tab θα χρειαστούμε και τα animator και animation. Για να τα δούμε πατάμε Window, τα επιλέγουμε και τα κάνουμε drag & drop όπου θέλουμε. Γενικά όλα τα tabs μπορούμε να τα μεταφέρουμε όπου μας βολεύουν για να φτιάξουμε το δικό μας workspace

3.2.2 Εισαγωγή Asset στο Workspace

Τα Assets πού θα χρησιμοποιήσουμε για το παιχνίδι είναι από το http://kenney.itch.io/kenney-donation τα οποία είναι public domain δηλαδή μπορούμε να τα χρησιμοποιήσουμε σε ότι είδους εφαρμογή θέλουμε. Για να τα βάλουμε μέσα στην unity, απλά κάνουμε drag & drop τον φάκελο στο Project Tab. Όλες οι εικόνες στον φάκελο έχουν γίνει αυτόματα sprite, και έτσι μπορούμε να τις κάνουμε drag & drop κατευθείαν στο scene. Βάζουμε ένα sprite μπροστά από την κάμερα, στον inspector στο scale βάζουμε την τιμή 1.3 στο X και Y για να φαίνετε το sprite καλύτερα και πατάμε Play. Αυτό το sprite θα είναι ο παίχτης μας



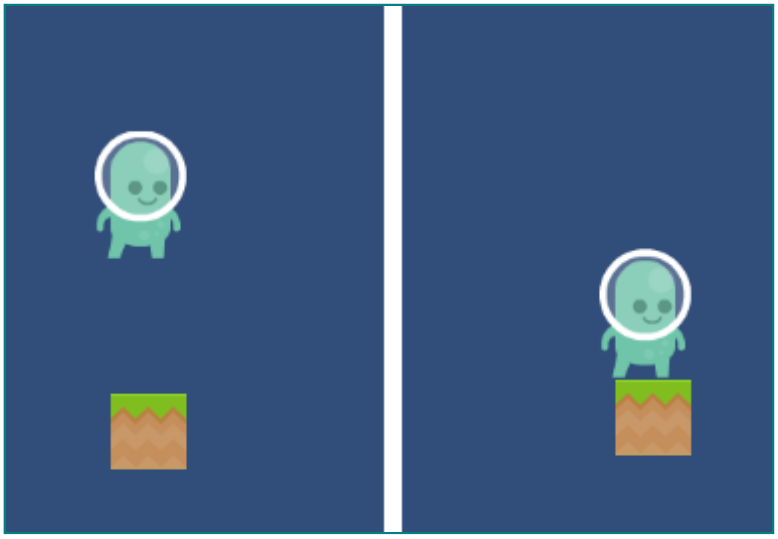
Εικόνα 3.2 : Το Workspace της Unity

3.2.3 GameObjects

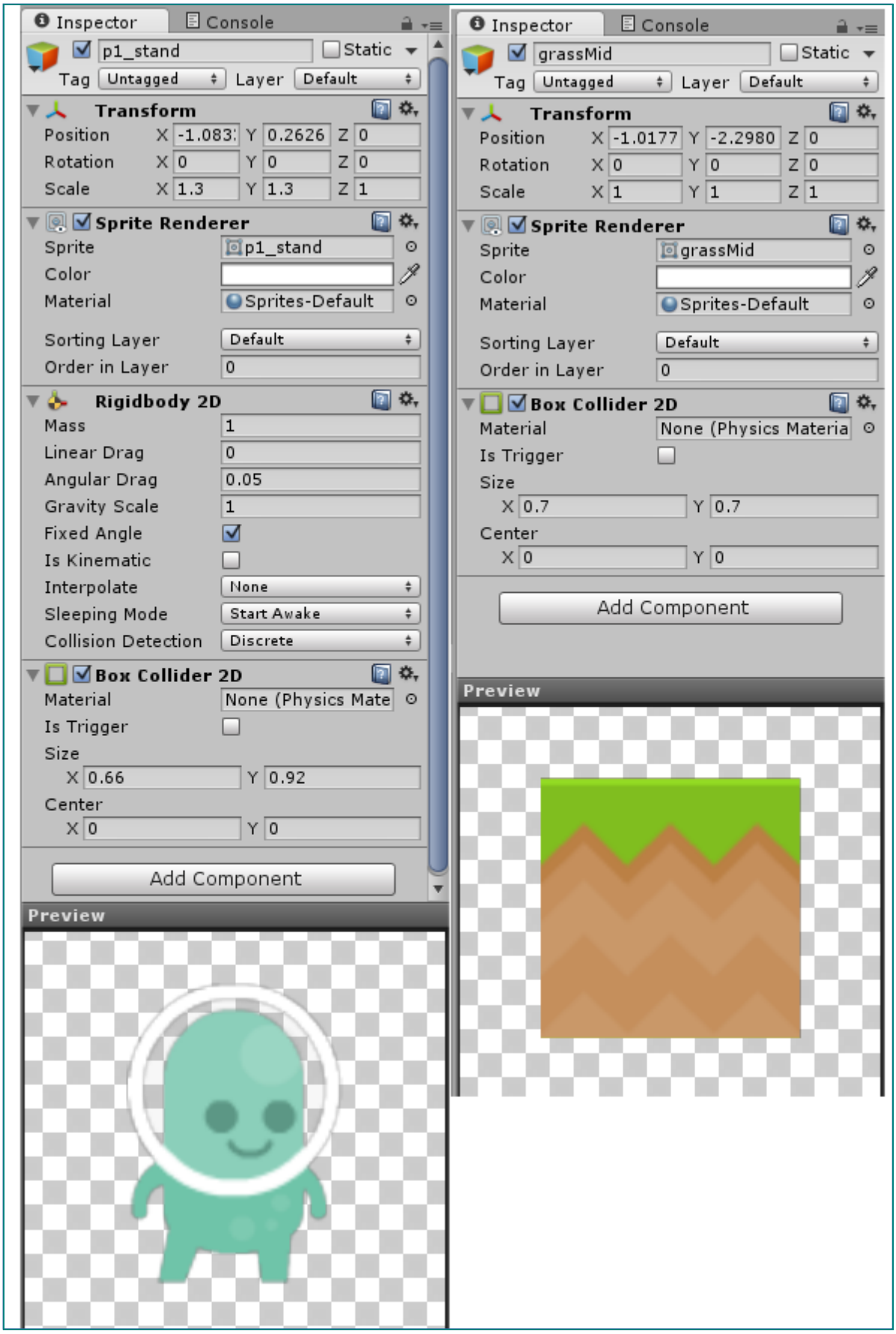
Τα GameObjects είναι οντότητες οι οποίες περιέχουν μέσα τους όλα τα components ενός αντικειμένου. Για παράδειγμα το sprite που βάλαμε στην σκηνή μας ποιό πάνω είναι ένα GameObject το οποίο έχει το component Transform (με το οποίο διαχειριζόμαστε την τοποθέτηση του αντικειμένου στον χώρο) και το component Sprite Renderer(το οποίο "ζωγραφίζει" το sprite στον χώρο). Πατώντας το κουμπί "Add Component" βλέπουμε όλα τα components που μπορούμε να προσθέσουμε σε ένα GameObject, επιλέγουμε Physics2D και Rigidbody 2D. Το Rigidbody 2D είναι ένα component το οποίο "λέει" στην μηχανή φυσικής της unity, να εφαρμόζει φυσική πάνω στο αντικείμενο. Αν πατήσουμε το κουμπί play, παρατηρούμε ότι τώρα επιδράει βαρύτητα πάνω στο σώμα, και πέφτει στο κενό. Αλλάζοντας τις μεταβλητές του Rigidbody, αλλάζουμε το πόσο

δυνατή είναι η βαρύτητα για αυτό το σώμα, την τριβή και άλλες μεταβλητές της μηχανής φυσικής, έτσι ώστε να φτιάξουμε το gameplay όπως θέλουμε.

Πηγαίνουμε πάλι στο Add Component , physics2D και επιλέγουμε Box Collider 2D. Ο box collider ελέγχει σε κάθε frame αν υπάρχει σύγκρουσή με κάποιον άλλο collider, και η μηχανή φυσικής την διαχειρίζεται. Κάνουμε Drag & Drop ένα sprite για πάτωμα, προσθέτουμε και σε αυτό ένα box collider 2D και το τοποθετούμε κάτω από το προηγούμενο sprite. Παρατηρούμε ότι το ένα sprite πέφτει πάνω στο άλλο και το πρώτο sprite σταματάει να πέφτει. Αφού το δεύτερο sprite δεν είναι rigidbody, αγνοεί τις δυνάμεις από φυσική ή συγκρούσεις. Έτσι θα φτιάξουμε τις πλατφόρμες στο παιχνίδι.



Εικόνα 3.3: Η επίδραση του Rigidbody2D στο GameObject



Εικόνα 3.4: GameObject Inspector

Πολλές φορές στην διάρκεια του development θα χρειαστεί να φτιάξουμε GameObjects που έχουν τα ίδια components με τις ίδιες μεταβλητές. Για να μην χρειαστεί να τα φτιάχνουμε κάθε φορά από την αρχή, αν κάνουμε drag&drop ένα GameObject από το Hierarchy στο tab Project, δημιουργούμε ένα prefab. Prefabs είναι GameObjects που κρατάνε αποθηκευμένα τα components (και τις μεταβλητές των components) που τους δώσαμε.

3.3 Συγγραφή σεναρίων για τα GameObjects

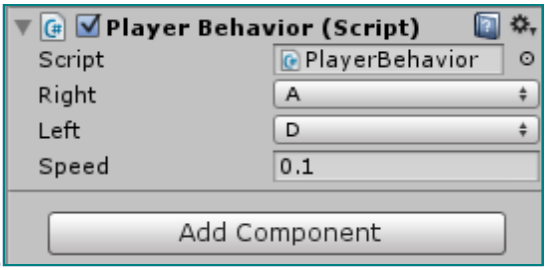
Για να μπορέσουμε να φτιάξουμε ένα παιχνίδι, πρέπει να μπορούμε να προγραμματίσουμε συμπεριφορά σε αντικείμενα, όπως AI σε αντιπάλους ή τι συμβαίνει όταν πατάμε ένα κουμπί σε κάποιο menu. Στην unity αυτό το πετυχαίνουμε μέσω scripts τα οποία προσθέτουμε σε GameObjects. Μπορούμε να γράψουμε scripts με JavaScript, C# ή Boo(η οποία έχει πολύ λιγότερο documentation από τις άλλες δύο και για αυτό δεν συνιστάτε). Η Unity συμπεριλαμβάνει το πρόγραμμα monodevelop για το γράψιμο scripts.

Για να προσθέσουμε ένα script στον παίχτη, πατάμε Add Component , new script, το ονομάζουμε PlayerBehavior και πατάμε create. Βλέπουμε ότι έχει δημιουργηθεί ένα script στον φάκελο Assets. Πατώντας διπλό κλικ το script ανοίγει με το monodevelop. Παρατηρούμε ότι το Script που φτιάξαμε έχει μόνο δύο άδεια functions. Το Start και το Update. Το function Start τρέχει μια φορά όταν δημιουργείτε το GameObject, και το function update τρέχει μια φορά σε κάθε frame. Δηλαδή η unity όταν ζωγραφίζει ένα frame "ζητάει" από κάθε Script να τρέξει την function update του. Χρησιμοποιώντας το function update θα δώσουμε κίνηση στον παίχτη. Το παιχνίδι θα παίζει κάπου στα 60 frames per second, έτσι αν πούμε στην function update "Αν ο παίχτης πατάει το κουμπί D πήγαινε 0.1 δεξιά", αν ο παίχτης έχει πατημένο το κουμπί D για ένα δευτερόλεπτο, ο παίχτης θα κινηθεί 6 δεξιά και η κίνηση θα φαίνετε smooth. Ακολουθεί το script για την κίνηση

```
public KeyCode right;

public KeyCode left;
public float speed;
// Update is called once per frame
void Update () {
	if (Input.GetKey (right))
		transform.position = new Vector2 (transform.position.x - speed, transform.position.y);
	if (Input.GetKey (left))
		transform.position = new Vector2 (transform.position.x + speed, transform.position.y);
}
```

Οι public μεταβλητές στα script μπορούν να αλλάξουν από τον Inspector. Αυτό βοηθάει στο να βρεις ακριβός τι τιμές θες για μεταβλητές όπως ταχύτητα, ύψος άλματος κ.α



Εικόνα 3.5: Public μεταβλητές στον Inspector

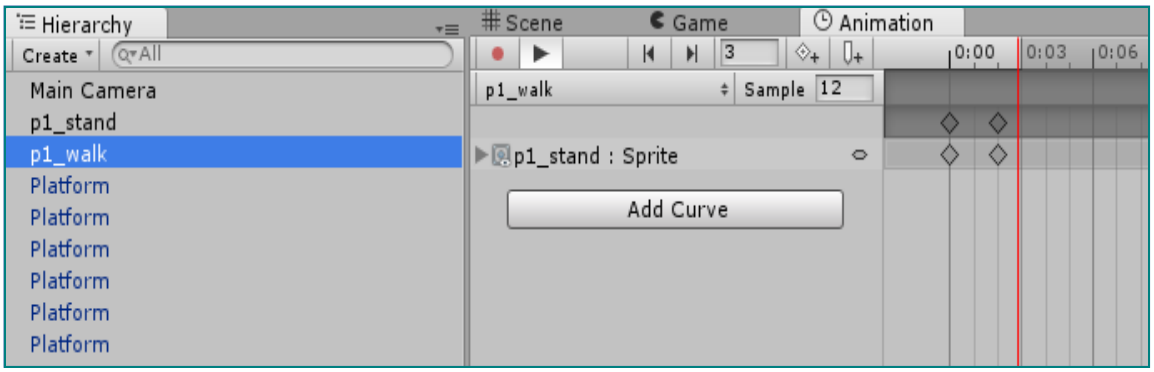
Κάποια άλλα πολύ σημαντικά function είναι τα OnCollisionEnter2D, OnCollisionStay2D και OnCollisionExit2D. Το OnCollisionEnter2D τρέχει μία φορά όταν γίνετε κάποιο collision με ένα άλλο collider, το OnCollisionStay2D τρέχει μια φορά σε κάθε frame στο οποίο υπάρχει collision με ένα άλλο collider, και το OnCollisionExit2D τρέχει μια φορά όταν σταματάει το collision με κάποιον άλλο collider. Αυτά τα function είναι πολύ χρήσιμα για να δώσουμε behaviors στα αντικείμενα μας , π.χ. ο παίχτης αν ακουμπήσει έναν αντίπαλο, θα χάσει μια καρδία, ή αν ακουμπήσει το νερό πεθαίνει .Υπάρχουν αντίστοιχα function για 3D collisions, αλλά δεν μας χρειάζονται. Ακολουθεί ένα παράδειγμα αυτών των function

```
void OnCollisionEnter2D(Collision2D coll) {
        if(coll.gameObject.name=="Platform")
                print ("Ksekinisa na akoubao tin platforma");
}
void OnCollisionStay2D(Collision2D coll) {
        if(coll.gameObject.name=="Platform")
                print ("Akoubao tin platforma");
}
void OnCollisionExit2D(Collision2D coll) {
        if(coll.gameObject.name=="Platform")
                print ("Stamatisa na akoubao tin platforma");
}
```

Αν το GameObject της πλατφόρμας έχει το όνομα "Platform", το tab console θα εκτυπώνει το αντίστοιχο μήνυμα.

3.4 Κίνηση (animation) των sprites

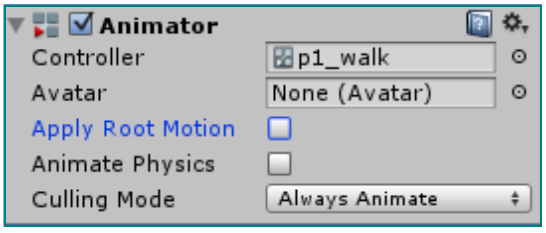
Τώρα που έχουμε προσθέσει βασική κίνηση στον παίχτη, θα προσθέσουμε ένα animation τρεξίματος. Η Unity έχει απλοποιήσει πολύ την δημιουργία και διαχείριση animations με sprites. Αν επιλέξουμε παραπάνω από ένα sprite και τα κάνουμε drag&drop στην σκηνή, θα πεταχτεί ένα μήνυμα για να φτιάξουμε animation με αυτά τα sprite, και έναν animator controller. Ας ονομάσουμε το animation P1_walk, ας το επιλέξουμε και πάμε στο animation tab



Εικόνα 3.6: Το animation tab

Στο animation tab βλέπουμε πόσα sprites έχει το animation, μπορούμε να αλλάξουμε την σειρά με την οποία παίζουν, και να αλλάξουμε την ταχύτητα πού τρέχουν (με το sample).

Τώρα που έχουμε φτιάξει το animation που θέλουμε, πρέπει να το προσθέσουμε στον παίχτη. Επειδή ώμος το walking animation δεν θα είναι το μόνο animation που θα έχει ο παίχτης, θα προσθέσουμε ένα component το οποίο θα ελέγχει ποιό animation πρέπει να παίζει σε κάθε στιγμή. Αυτό το component είναι το animator.Πηγαίνουμε στον παίχτη, πατάμε add component , Miscellaneous και animator. Στο animator component, ξε-επιλέγουμε το apply root motion, και στο controller διαλέγουμε το P1_walk

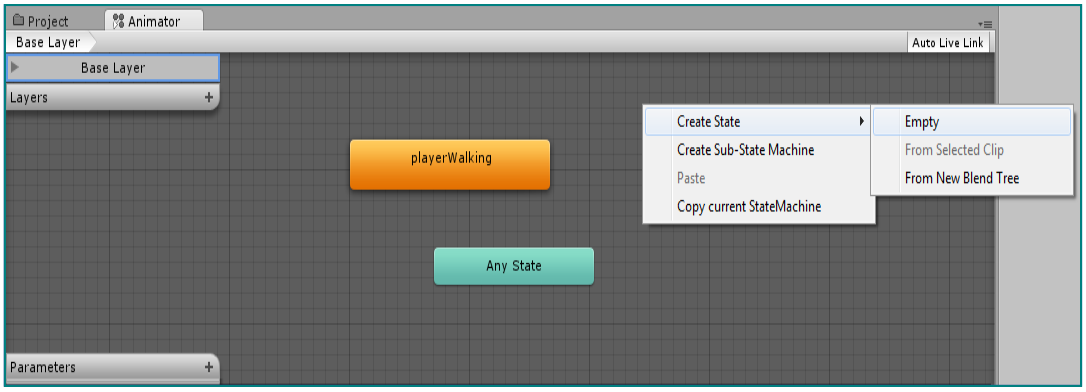


Εικόνα 3.7: Το animator component

Τώρα πηγαίνουμε στο animator tab. Το animator είναι ένας διαχειριστής animation. Έχει διάφορες μεταβλητές που λέγονται parameters, animation states

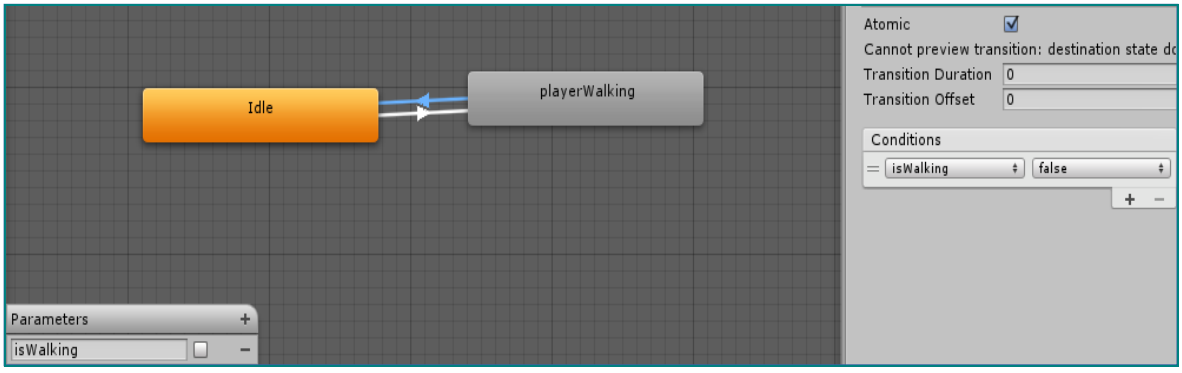
τα οποία έχουν ένα animation μέσα τους το καθένα, και transitions, τα οποία αλλάζουν το current state ανάλογα με τα parameters. π.χ. αν το parameter bool walk το οποίο παίρνει την είναι true, τότε κάνε transition από το idle state στο walking state. Το walking state θα έχει το walking animation και το idle state δεν θα έχει κανένα animation. Το Idle state θα είναι το default state, δηλαδή το "διάγραμμα" θα ξεκινάει από αυτό.

Για αρχή φτιάχνουμε ένα νέο state και το ονομάζουμε Idle



Εικόνα 3.8: Το animator tab

Πατάμε δεξί κλικ στο Idle και πατάμε Set Default. Μετά πατάμε κλικ στο + δίπλα από το Parameters, επιλέγουμε bool. Ονομάζουμε την παράμετρο isWalking. Μετά πατάμε δεξί κλικ στο Idle, make transition και επιλέγουμε το walking. Πατάμε στο transition που δημιουργήθηκε, και στα conditions επιλέγουμε isWalking και true. Έτσι αν η παράμετρος isWalking γίνει true, θα πάμε στο walking state από το Idle state. Πατάμε τώρα δεξί κλικ στο walking state, make transition και διαλέγουμε το Idle. Στο δεύτερο transition βάζουμε το condition isWalking false, έτσι ώστε αν το isWalking γίνει false, να πάμε στο Idle state



Εικόνα 3.9: Transitions ανάμεσα σε animation

Τώρα αυτό που μένει να κάνουμε είναι να δίνουμε την τιμή true στο isWalking όταν

ο παίχτης πατάει το κουμπί A ή D και false όταν σταματάει να τα πατάει. Αυτό το κάνουμε στο script του παίχτη. Στην function start προσθέτουμε:

```
Animator characterAnim;
void Start () {
	characterAnim = GetComponent<Animator>();
}
```

Έτσι η μεταβλητή characterAnim δείχνει στον animator του παίχτη. Μετά η update function γίνετε:

```
void Update ()
{
	if (Input.GetKey (right))
		transform.position = new Vector2 (transform.position.x - speed, transform.position.y);
	if (Input.GetKey (left))
		transform.position = new Vector2 (transform.position.x + speed, transform.position.y);
	if ((Input.GetKey (right)) || (Input.GetKey (left)))
		characterAnim.SetBool ("isWalking", true);
	else
		characterAnim.SetBool ("isWalking", false);

}
```

Ένα πρόβλημα που δημιουργείτε είναι ότι το sprite του παίχτη δεν γυρίζει όταν ο παίχτης αλλάζει κατεύθυνση. Κανονικά θα χρειαζόταν να φτιάχναμε flipped sprites και animations για αυτό, αλλά με την unity μπορούμε αν πολλαπλασιάσουμε την τιμή scale.x με το -1, να κάνουμε αυτόματα flip τα sprites και τα animations όταν πρέπει. Έτσι προσθέτοντας το function Flip() , το τελικό script του παίχτη είναι:

```
public class PlayerBehavior : MonoBehaviour {
	Animator characterAnim;
	void Start () {
		characterAnim = GetComponent<Animator>();
	}
	public KeyCode right;
	public KeyCode left;
	public float speed;
	bool isFacingRight=true;



	void Update () {

		if (Input.GetKey (right)){
			transform.position = new Vector2 (transform.position.x + speed, transform.position.y);
				if(isFacingRight==false)
					Flip();
		}
		if (Input.GetKey (left)){
			transform.position = new Vector2 (transform.position.x - speed,transform.position.y);
				if(isFacingRight==true)
					Flip();
		}
		if ((Input.GetKey (right)) || (Input.GetKey (left)))
			characterAnim.SetBool ("isWalking", true);
		else
			characterAnim.SetBool ("isWalking", false);
	}

	void Flip(){
		transform.localScale = new Vector2 (transform.localScale.x * -1, transform.localScale.y);
		if (isFacingRight)
			isFacingRight = false;
		else
			isFacingRight=true;
	}

	void OnCollisionEnter2D(Collision2D coll) {
		if(coll.gameObject.name=="Platform")
			print ("Ksekinisa na akoubao tin platforma");
	}
	void OnCollisionStay2D(Collision2D coll) {
		if(coll.gameObject.name=="Platform")
			print ("Akoubao tin platforma");
	}
	void OnCollisionExit2D(Collision2D coll) {
		if(coll.gameObject.name=="Platform")
			print ("Stamatisa na akoubao tin platforma");
	}
}
```

3.5 Ευφυΐα των αντικειμένων

Με την παραπάνω λογική μπορούμε να φτιάξουμε λογική στα αντικείμενα για το πως αντιδρούν στον παίχτη. Για παράδειγμα αν ο παίχτης ακουμπήσει ένα κέρμα, το κέρμα εξαφανίζετε και προστίθενται 10 πόντοι στο σκορ του. Αυτό μπορούμε να το κάνουμε εύκολα με τις OnCollision functions που είδαμε παραπάνω. Συχνά όμως χρειαζόμαστε ποιό πολύπλοκη λογική σε gameobjects, για παράδειγμα ένας αντίπαλος ο οποίος προχωράει δεξιά και αριστερά, αλλά όταν ο παίχτης μπει στο βεληνεκές του σταματάει να κινείτε και αρχίζει να το πυροβολεί. Υπάρχουν κάποιες λειτουργίες της Unity που μπορούν να μας βοηθήσουν με αυτό. Στην κλάση Physics2D υπάρχουν τα functions OverlapCircle και OverlapArea, οι οποίες δημιουργούν ένα κύκλο ή ένα τετράγωνο αντίστοιχα, σε κάποιο σημείο που θέλουμε , και επιστρέφουν τον collider2D που υπάρχει μέσα σε αυτές τις περιοχές, αν υπάρχει. Μετά με μια απλή if λέμε στο GameObject ότι αν το name του collider2D που βρήκε είναι "Player", να εκτελέσει τις πράξεις που θέλουμε. Έτσι μπορούμε να φτιάχνουμε εύκολα Line of Sight για αντιπάλους

Ένα άλλο χρήσιμο function για την δημιουργία AI είναι το SendMessage της κλάσης GameObject. Αυτό το function λέει στο GameObject που στοχεύει να εκτελέσει κάποιο function, το όνομα του οποίου δίνετε στις παραμέτρους της εντολής. Για παράδειγμα μια σφαίρα έχει τον εξής κώδικα

```
void OnCollisionEnter2D(Collision2D target) {
        if(target.gameObject.name=="Player")
                target.gameObject.SendMessage ("gotHit", 5f);
}
```

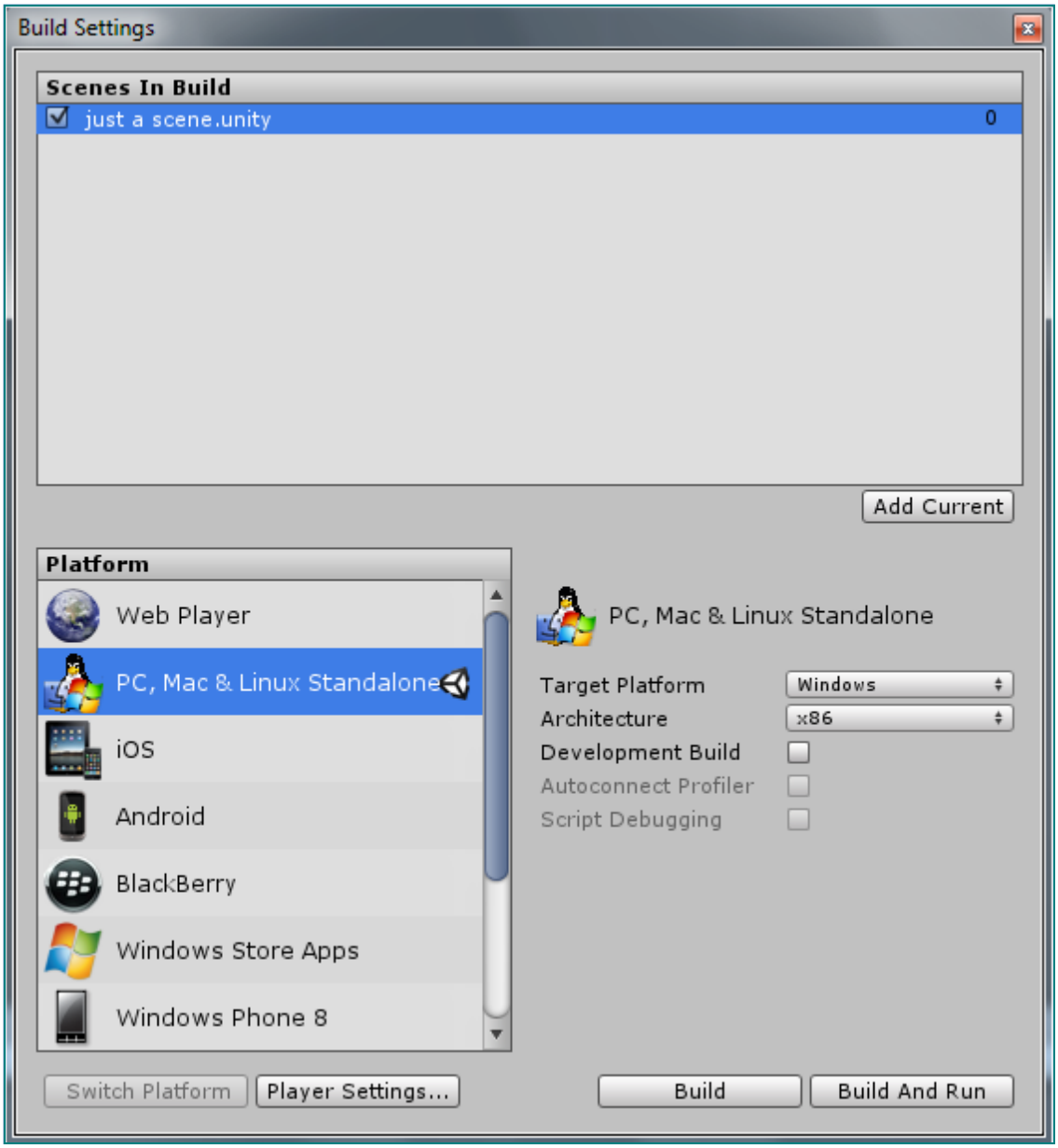
Με τον παραπάνω κώδικα, όταν μια σφαίρα πετυχαίνει τον παίχτη, του λέει να τρέξει το function gotHit(float damage) που έχει, με παράμετρο damage 5.

3.6 Σύστημα αποθήκευσης προόδου

Ένα άλλο πολύ σημαντικό μέρος της ανάπτυξης παιχνιδιών είναι το σύστημα αποθήκευσης δεδομένων. Δεδομένα όπως το score του παίχτη, το τελευταίο checkpoint που ακούμπησε κ.α. πρέπει να αποθηκεύονται , και να μπορούν να ανακτηθούν όταν ο παίχτης ξεκινάει το παιχνίδι. Η Unity βοηθάει σε αυτό τον τομέα με την κλάση PlayerPrefs. Η κλάση PlayerPrefs έχει μέσα τα functions SetInt, SetFloat και SetString για να αποθηκεύουμε μεταβλητές στο registry, στην τοποθεσία HKCU\Software\[company name]\[product name] key όπου τα company και product name τα δίνουμε στο Project Settings, και το key το δίνουμε όταν καλούμε τα functions. π.χ. αν δώσουμε την εντολή PlayerPrefs.SetInt("Speed",10); και μετά την εντολή Save(); η unity θα δημιουργήσει το κλειδί Speed που είναι int, και θα βάλει την τιμή 10 μέσα του. Με τις function GetInt, GetString και GetFloat μπορούμε να πάρουμε τα δεδομένα που είναι αποθηκευμένα σε κάποιο κλειδί.

3.7 Τα επίπεδα και η δημιουργία του προγράμματος

Τώρα που έχουμε ένα πολύ βασικό πρόγραμμα, θα δούμε πως μπορούμε να το κάνουμε build για να παίζει εκτός της Unity. Η Unity μας δίνει μεγάλη εμβέλεια στις επιλογές για την πλατφόρμα για την οποία θα φτιάξουμε το παιχνίδι μας. Χωρίς καθόλου extra δουλειά, το παιχνίδι μας μπορεί να παίξει σε Windows, Mac, Linux , Ps3, Wii, Xbox360, IOS,Android,Windows Phone ακόμα και κατευθείαν σε Browser. Για να φτιάξουμε το πρόγραμμα πατάμε File και Build Settings. Πατάμε το κουμπί Add Current, για να προσθέσουμε την σκηνή που φτιάξαμε στο Build. Όταν φτιάξουμε παραπάνω σκηνές/level, θα κάνουμε την κάθε μια add, και μετά μέσα από script με την εντολή Application.LoadLevel θα αλλάζουμε levels όταν χρειάζεται.



Εικόνα 3.10: Οι επιλογές του build

Αφού διαλέξουμε την πλατφόρμα, πατάμε το κουμπί Build And Run, ονομάζουμε το αρχείο και το τρέχουμε

3.8 Συμπεράσματα

Η Unity είναι ένα πολύ δυνατό και εύχρηστο εργαλείο για την δημιουργία 3D και 2D παιχνιδιών. Με την ευχρηστία και απλότητα του editor της κάνει την unity αρκετά απλή για να μάθει κάποιος game design με αυτή, αλλά το πλήθος των δυνατοτήτων που μας προσφέρει της δίνουν αρκετό βάθος ώστε να είναι ανταγωνιστική σε σχέση με άλλες μηχανές γραφικών.

Η θυσία που κάνει η Unity για να είναι ανταγωνιστική σαν μια game engine γενικής χρήσης είναι ότι game engines ειδικευμένες σε κάποιο σκοπό είναι συχνά καλύτερες επιλογές για τον συγκεκριμένο σκοπό, όπως για παράδειγμα το Adventure Game Studio είναι καλύτερη επιλογή αν θες να αναπτύξεις ένα Adventure Game. Ένα άλλο παράδειγμα αυτού είναι ότι επειδή δεν είναι βελτιστοποιημένη για mobile games, τα παιχνίδια για κινητά που φτιάχνονται με την Unity, τείνουν να είναι ποιό βαριά από άλλες μηχανές, και μέχρι το 4ο version της Unity χρησιμοποιούσαν μόνο τον ένα πύρινα, κάτι που είναι πολύ αρνητικό αν σκεφτείς ότι τα κινητά κατά κύριο λόγο δεν είναι αρκετά δυνατά μηχανήματα.

Πέρα από αυτό η Unity είναι ιδανική για μικρές εταιρίες που δεν έχουν το budget για να φτιάξουν μια custom μηχανή για τις ανάγκες τους, αλλά και για μαθητές game design και προγραμματισμού που αρκούνται στο free version της.

Βιβλιογραφία

[1] Mark J. P. Wolf (2008). The video game explosion: a history from PONG to Playstation and beyond.

[2] Bates, Bob (2004). *Game Design* (2nd ed.).

[3] Bethke, Erik (2003). Game development and production.

[4] http://www.techdirt.com/articles/20120129/17272817580/sky-is-rising-entertainment-industry-is-large-growing-not-shrinking.shtml