Αχιλλέας Παπασταματίου ΑΜ: 00070

Οι νεκροί ακούνε τζαζ!

Από ότι φαίνεται οι γλίτσες δε σέβονται ούτε νεκρούς ούτε ζωντανούς. Δεν μπορεί ένας σκελετός δηλαδή να ακούσει το σαξόφωνό του με την ησυχία του; Ας είναι... Πάρε το σπαθί σου, τράβα το αγαπημένο σου ζόμπι από το χέρι και πάμε να σκοτώσουμε γλίτσα!

Το «*Οι νεκροί ακούνε τζαζ*!» (Αγγλικός τίτλος: "*Jazz for the dead!*") είναι ένα co-op, fighting, top-down παιχνίδι για 2 άτομα, με περιορισμένο αριθμό επιθέσεων ανά παίχτη. Η ομάδα μοιράζεται τις ίδιες ζωές - είτε κερδίζεις σαν ομάδα, είτε χάνεις σαν ομάδα. Σκοπός είναι η σφαγή της περισσότερης δυνατής γλίτσας για την επίτευξη του υψηλότερου δυνατού σκορ.

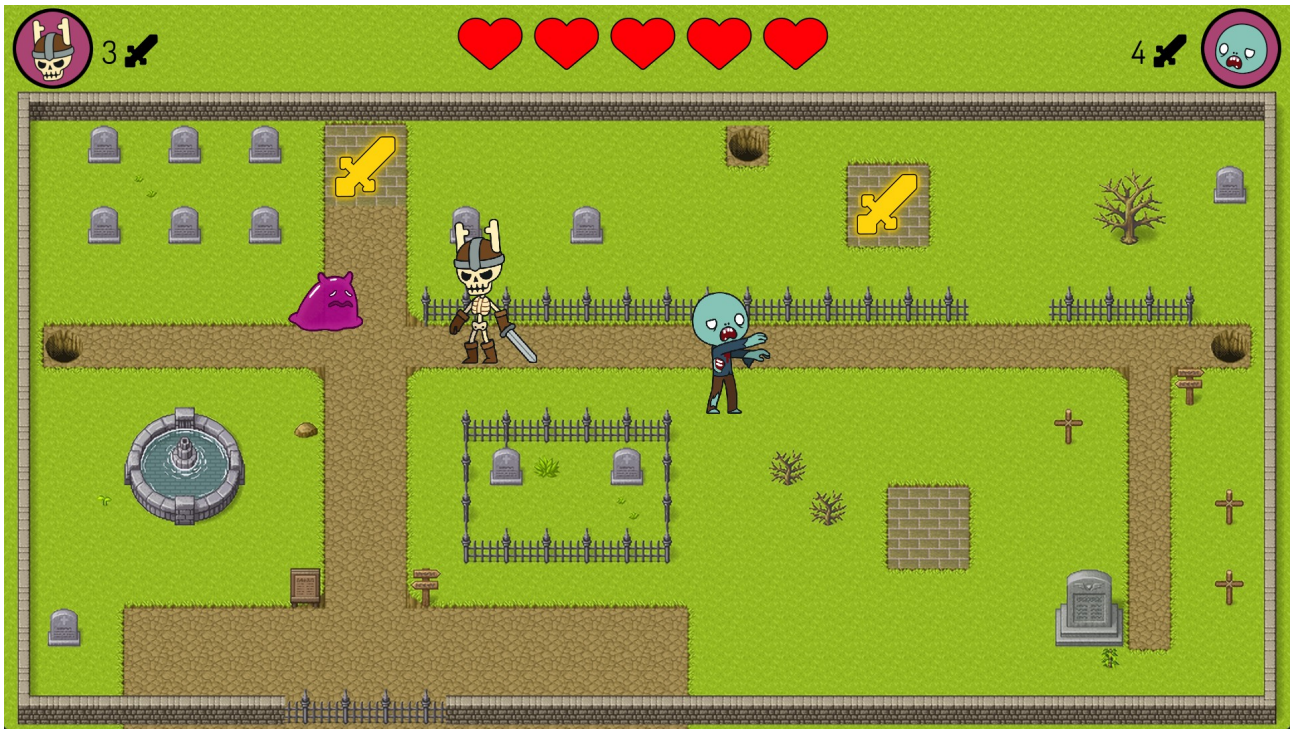
Game design & mechanics

Το παιχνίδι ακολουθεί τη λογική του top-down με κίνηση των παιχτών σε δύο άξονες και με απλό σύστημα επιθέσεων βασισμένο στα collisions μεταξύ παιχτών και εχθρών. Όλες οι επιθέσεις είναι melee. Ο κάθε παίχτης μπορεί να κινηθεί πάνω-κάτω και αριστερά-δεξιά με τα βελάκια ή με τα πλήκτρα WASD και να επιτεθεί με το Space. Αν ένας παίχτης αγγίξει έναν εχθρό (μία γλίτσα) όσο περπατάει ή είναι ακίνητος, ο εχθρός του αφαιρεί μία ζωή. Αν όμως ο παίχτης βρισκόταν σε κατάσταση επίθεσης, δηλαδή κατά τη διάρκεια του animation της επίθεσης, ο εχθρός πεθαίνει.

Τα δύο χαρακτηριστικά του gameplay που κάνουν το παιχνίδι να ξεχωρίζει είναι η διαχείριση των ζωών και των επιθέσεων. Ο αριθμός των ζωών είναι κοινός για τους δύο παίχτες. Η ομάδα ξεκινάει με 5 ζωές σε κάθε επίπεδο και κάθε φορά που ένας εχθρός καταφέρνει να αγγίξει οποιονδήποτε ευάλωτο παίχτη, η ομάδα χάνει μία ζωή. Αν η ομάδα χάσει και τις 5 ζωές, το παιχνίδι τελειώνει.

Κάθε παίχτης ξεκινάει με έναν πεπερασμένο αριθμό διαθέσιμων επιθέσεων (σπαθιών). Κάθε επίθεση μειώνει τα διαθέσιμα σπαθιά αυτού του παίχτη κατά 1, σαν να ήταν σφαίρες, ασχέτως του αν η επίθεση σκότωσε εχθρό ή ήταν άστοχη. Μέσα στο επίπεδο εμφανίζονται επιπλέον σπαθιά που αν τα αγγίξει κάποιος παίχτης, αποκτά μία ακόμα επίθεση.

Στόχος του game design είναι η ενθάρρυνση των παιχτών να συνεργαστούν, τόσο λεκτικά όσο και αυθόρμητα, αλλά και να διαχειριστούν έναν περιορισμένο πόρο με τον βέλτιστο τρόπο. Στην προκειμένη περίπτωση ο πόρος συμβολίζει την επιθετικότητα. Το gameplay εστιάζει στη διαχείριση των διόδων διαφυγής, καθώς ένας περικυκλωμένος παίχτης δεν έχει τη δυνατότητα να σκοτώσει άμεσα όλες τις βραχυπρόθεσμες απειλές. Επίσης, η θέση του κάθε παίχτη στη σκηνή θα επηρεάσει την πιθανότητα αποφυγής εχθρών μακροπρόθεσμα. Έτσι δημιουργούνται ευκαιρίες για τους παίχτες να διαμορφώσουν τη δική τους στρατηγική.



Επίπεδα

Το παιχνίδι αποτελείται από 2 επίπεδα δυσκολίας. Για να ολοκληρωθεί επιτυχώς ένα επίπεδο, πρέπει η ομάδα να σκοτώσει όλους τους εχθρούς του. Κάθε επίπεδο έχει τη δική του σκηνή και μουσική. Στο πρώτο επίπεδο κάθε παίχτης ξεκινάει με 4 διαθέσιμες επιθέσεις και 10 εχθροί θα εμφανιστούν κατά τη διάρκεια του παιχνιδιού. Στο δεύτερο επίπεδο, οι διαθέσιμες επιθέσεις είναι 3 ανά παίχτη και οι εχθροί είναι 15. Κάθε επίπεδο έχει 3 σημεία της σκηνής στα οποία μπορούν να εμφανιστούν σπαθιά και 3 σημεία από τα οποία εμφανίζονται οι εχθροί. Οι ζωές της ομάδας ξαναγεμίζουν στις 5 ανάμεσα στα επίπεδα. Αν μία ομάδα ολοκληρώσει επιτυχώς και τα δύο επίπεδα, κερδίζει το παιχνίδι, αλλά το σκορ της ακόμη εξαρτάται και από το πόσες ζωές έχασε.

Βαθμολόγηση

Η ομάδα των δύο παιχτών συγκεντρώνει πόντους συνεργατικά. Κάθε ηττημένος εχθρός χαρίζει άμεσα 10 πόντους. Στο τέλος κάθε επιπέδου που ολοκληρώθηκε επιτυχώς, οι υπολειπόμενες ζωές προσφέρουν 20 πόντους η κάθε μία. Για παράδειγμα, η ομάδα *Dawn of the Dev* σκότωσε όλες τις 10 γλίτσες του πρώτου επιπέδου και επιβίωσε με 2 ζωές, αλλά πέθανε στο επόμενο επίπεδο, έχοντας σκοτώσει άλλες 3 γλίτσες. Το σκορ της ήταν ((10 * 10) + (2 * 20)) + (3 * 10) = 170 πόντοι.

Το μέγιστο δυνατό σκορ, για άψογη ολοκλήρωση και των δύο επιπέδων, θα είναι ((10 * 10) + (5 * 20)) + ((15 * 10) + (5 * 20)) = 450 πόντοι.

Ο χρόνος ολοκλήρωσης κάθε επιπέδου, η συνεισφορά του κάθε παίχτη στην ομάδα του και το πλήθος των επιπλέον επιθέσεων που μαζεύτηκαν δεν επηρεάζουν το σκορ.

Δομή του κώδικα

Ο κώδικας είναι μοιρασμένος σε 3 αρχεία python και υλοποιεί μία αρχιτεκτονική server-client για τη multiplayer λειτουργικότητα. Το *jazz_server.py* και το *jazz_client.py* τρέχουν τις διεργασίες των δύο παιχτών και το *jazz_operations.py* χρησιμοποιείται σαν βιβλιοθήκη συναρτήσεων και σταθερών, για εύκολη παραμετροποίηση. Το *jazz_operations.py* εισάγεται στα άλλα δύο αρχεία ως *jo* για συντομία στις κλήσεις. Ο server και ο client ξεκινάνε και οι δύο με μία αρχικοποίηση μεταβλητών, συνδέσεων και στοιχείων της pygame και μετά ακολουθεί το μεγάλο, βασικό loop της pygame, μέσα στο οποίο γίνεται η διαχείριση του input του χρήστη, το rendering καθενός frame του παιχνιδιού και οποιοσδήποτε άλλος υπολογισμός είναι απαραίτητος για το τρέχον frame. Ο server είναι υπεύθυνος για κάποιες λειτουργίες που δε χρειάζεται να εκτελεστούν και από τις δύο πλευρές, όπως θα δούμε παρακάτω.

Υπάρχουν 6 διαφορετικές οθόνες που μπορούν να γίνουν render (και στον server και στον client). Η μεταβλητή menu_screen διατηρεί την τιμή του τρέχοντος τύπου λειτουργίας και αναλόγως γίνονται render τα στοιχεία που πρέπει. Μέσα σε κάθε τύπο οθόνης, διάφορα boolean flags συνεισφέρουν στην επιλογή των ορατών στοιχείων. Η menu_screen παίρνει θετικές τιμές για οθόνες των μενού και αρνητικές για πραγματικό gameplay, συγκεκριμένα:

- ‘-1’ για gameplay, το οποίο ανήκει σε ένα από τα δύο επίπεδα και περιλαμβάνει τη σκηνή, τους δύο χαρακτήρες, τους εχθρούς, τα σπαθιά και το HUD με πληροφορίες για τις ζωές και τις διαθέσιμες επιθέσεις,
- ‘1’ για την πρώτη οθόνη του μενού, όπου ο client συνδέεται με τον server,
- ‘2’ για την επόμενη οθόνη του μενού, όπου ο server εισάγει ένα όνομα για την ομάδα,
- ‘3’ για την οθόνη εκκίνησης του παιχνιδιού, όπου οι παίχτες μαθαίνουν ποιον χαρακτήρα παίζει ο καθένας,
- ‘4’ για την οθόνη που εμφανίζεται ενδιάμεσα στα δύο επίπεδα,
- ‘5’ για το τέλος του παιχνιδιού, όπου εμφανίζεται ο πίνακας με τις καλύτερες ομάδες.

Στις οθόνες των μενού, υπάρχουν διαφορές ανάμεσα στο τι κάνει render ο server και τι ο client. Στην αρχική οθόνη, ο server δείχνει την IP του στο τοπικό δίκτυο και ακούει για συνδέσεις και ο client έχει ένα text field στο οποίο πρέπει να τη γράψει. Σε αυτό το σημείο ο client δέχεται input μόνο από κουμπιά που θα μπορούσαν να ανήκουν σε μία διεύθυνση IP.

```
14  # Keys that are allowed to be pressed during IP user input.
15  IP_CHARS = [pygame.K_0, pygame.K_1, pygame.K_2, pygame.K_3, pygame.K_4, pygame.K_5, pygame.K_6, pygame.K_7,
16              pygame.K_8, pygame.K_9, pygame.K_KP0, pygame.K_KP1, pygame.K_KP2, pygame.K_KP3, pygame.K_KP4,
17              pygame.K_KP5, pygame.K_KP6, pygame.K_KP7, pygame.K_KP8, pygame.K_KP9, pygame.K_PERIOD, pygame.K_KP_PERIOD]
```

Όταν ο χρήστης πατήσει Enter, η συνάρτηση is_ipv4(string) ελέγχει αν πρόκειται για έγκυρη διεύθυνση IP και μετά προσπαθεί να συνδεθεί.

```
1 usage
def is_ipv4(string):
    """ Return 'True' if the given string is a valid IPv4 address or 'False' otherwise."""
    try:
        IPv4Network(string)
        return True
    except ValueError:
        return False
```

Κάτω από το text field μπορούν να εμφανιστούν δύο πιθανά σφάλματα, ένα για μη έγκυρη διεύθυνση και ένα για αποτυχία σύνδεσης, το οποίο μπορεί να συμβεί αν η διεύθυνση είναι λάθος ή αν ο server δεν τρέχει.

```
        # Handle user input for the host IP.
        if event.type == pygame.KEYDOWN and input_active:
            if event.key == pygame.K_RETURN or event.key == pygame.K_KP_ENTER:
                input_active = False
                show_ip_error = False
                if is_ipv4(host_ip):
                    try:
                        s.connect((host_ip, jo.PORT))
                        # print("Connection established.")
                        menu_screen += 1
                    except socket.error:
                        ip_error_text = jo.dosis_font.render("Error: Host not found", 1, jo.PINK)
                        show_ip_error = True
                        input_active = True
                else:
                    ip_error_text = jo.dosis_font.render("Error: Invalid IP address", 1, jo.PINK)
                    show_ip_error = True
                    input_active = True
            elif event.key == pygame.K_BACKSPACE and len(host_ip) > 0:
                show_ip_error = False
                host_ip = host_ip[:-1]
            elif len(host_ip) < 15 and event.key in IP_CHARS:
                show_ip_error = False
                host_ip += event.unicode
```

Όταν επιτευχθεί σύνδεση, και ο server και ο client μεταφέρονται στην επόμενη οθόνη, δηλαδή η menu_screen γίνεται ‘2’. Εδώ εμφανίζεται ένα text field στον server για να γράψει ένα όνομα για την ομάδα. Η λογική των ελέγχων είναι παρόμοια, αλλά τώρα αποδεκτά είναι τα γράμματα και το κενό. Ο client δείχνει ότι ο συμπαίκτης του εισάγει όνομα και περιμένει μέχρι ο server να πατήσει Enter.

Μόλις ο server στείλει το όνομα της ομάδας στον client, και οι δύο μεταβαίνουν στην οθόνη εκκίνησης. Επιλέγεται τυχαία από τον server ο ρόλος του κάθε παίχτη, δηλαδή ποιος θα παίξει ως σκελετός και ποιος ως ζόμπι και η επιλογή στέλνεται στον client.

```
342         elif menu_screen == 3:      # Start screen.
343             if len(server_role) == 0:
344                 if random() < 0.5:
345                     server_role = "s"  # Server is playing skeleton.
346                     client_role = "z"  # Client is playing zombie.
347                 else:
348                     server_role = "z"  # Server is playing zombie.
349                     client_role = "s"  # Client is playing skeleton.
350                 try:
351                     conn.sendall(client_role.encode())
352                 except socket.error:
353                     print("Failed to send client role to client.")
354                     run = False
355             jo.draw_start_menu(team_name, screen, server_role)
```

Εδώ χρησιμοποιείται η συνάρτηση draw_start_menu(team_name, screen, role) για να προστεθούν τα διάφορα στοιχεία στην οθόνη, επειδή η μόνη διαφορά ανάμεσα στον server και στον client είναι το κουμπί Start που είναι εμφανές μόνο στον server. Όταν πατηθεί το κουμπί, ο server ενημερώνει τον client, η μουσική του μενού σβήνει με fade out και ξεκινάει μία αντίστροφη μέτρηση για την μετάβαση στο gameplay. Το κουμπί Start λειτουργεί με το κλικ του ποντικιού.

```
259         # Event listener for the start button.
260         if event.type == pygame.MOUSEBUTTONUP and (menu_screen == 3 or menu_screen == 4) and start_active:
261             mouse_pos = pygame.mouse.get_pos()
262             if start_button.get_rect(topleft=(jo.width_center - start_button.get_width() / 2, 780)). \
263                     collidepoint(mouse_pos):
264                 start_gameplay()
```

Η συνάρτηση countdown_from(seconds, screen) δείχνει και στους δύο παίχτες τα νούμερα από το ‘3’ μέχρι το ‘1’ και παίζει έναν χαρακτηριστικό ήχο ανά δευτερόλεπτο.

```
    4 usages
93  def countdown_from(seconds, screen):
94      """ Start a blocking countdown for the given amount of time, with visual and audio feedback.
95
96      Parameters:
97          seconds (int): Number of seconds to count down from.
98          screen (pygame.Surface): The caller's main pygame Surface.
99      """
100
101     countdown = seconds
102     while countdown > 0:
103         countdown_text = dosis_font_large.render(str(countdown),  antialias: 1, PINK)
104         countdown_text_rect = countdown_text.get_rect(center=(width_center, 780))
105         pygame.draw.rect(screen, BLACK,  rect: (width_center - 50, 780 - 50, 100, 100))    # Erase previous number.
106         screen.blit(countdown_text, countdown_text_rect)
107         pygame.display.update()
108         ding_sound.play()
109         pygame.time.delay(1000)
110         countdown -= 1
```

Επειδή γίνεται χρήση της pygame.time.delay(milliseconds), που είναι blocking, τα frames του παιχνιδιού δεν ανανεώνονται μέσω του βασικού loop, αλλά μέσα στην countdown_from(seconds, screen). Δηλαδή τα FPS του παιχνιδιού γίνονται προσωρινά ίσα με 1. Με το τέλος της αντίστροφης μέτρησης, η menu_screen γίνεται ‘-1’ και ξεκινάει το παιχνίδι.

Κατά τη διάρκεια του παιχνιδιού, η διεργασία του κάθε παίχτη ανταποκρίνεται στο input για μετακίνηση και επίθεση και παρακολουθεί τη θέση, τη φορά (αν κοιτάει αριστερά ή δεξιά), το πλήθος των διαθέσιμων επιθέσεων, την κατάσταση του χαρακτήρα (αδράνεια/περπάτημα/επίθεση), αλλά και τον δείκτη της λίστας εικόνων (στην πραγματικότητα αντικειμένων pygame.Surface) για

το τρέχον animation. Η διεργασία του κάθε παίχτη είναι υπεύθυνη για αυτές τις μεταβλητές μόνο για τον χαρακτήρα του παίχτη της.

Επιπλέον, ο server παρακολουθεί και το σκορ, το πλήθος ζωών της ομάδας, το πλήθος και τις θέσεις των σπαθιών στη σκηνή (αν υπάρχουν), το πλήθος, τις θέσεις και τους δείκτες των animation των εχθρών (αν υπάρχουν) και ακόμα ελέγχει αν το gameplay πρέπει να διακοπεί (νίκη ή ήττα). Τα στοιχεία των σπαθιών αποθηκεύονται στην λίστα swords[]. Κάθε στοιχείο της λίστας είναι απλά ένας δείκτης της θέσης σπαθιού του επιπέδου. Για παράδειγμα, η λίστα [2, 0] σημαίνει ότι υπάρχουν δύο ορατά σπαθιά αυτή τη στιγμή, ένα στην 1η και ένα στην 3η από τις 3 πιθανές θέσεις. Κάθε στιγμή μπορούν να είναι ενεργά από 0 έως 3 σπαθιά. Αντίστοιχα, τα στοιχεία των εχθρών αποθηκεύονται στη λίστα slimes[]. Κάθε στοιχείο της λίστας είναι ένα tuple που περιέχει τη θέση του εχθρού (tuple), τον δείκτη του animation του εχθρού (int) και το αν το γραφικό του εχθρού είναι αντεστραμμένο οριζόντια σε σχέση με το αρχείο εικόνας του sprite (boolean). Όλα τα sprites κοιτάνε αριστερά στα αρχεία png που εισάγονται στο παιχνίδι, οπότε όταν η boolean είναι ‘True’ σημαίνει ότι ο εχθρός κοιτάζει δεξιά. Για παράδειγμα, αν η λίστα slimes έχει τιμή:

[((150, 73), 4, True), ((412, 128), 2, False)]

σημαίνει ότι υπάρχουν δύο ζωντανοί εχθροί αυτή τη στιγμή, ο πρώτος βρίσκεται στη θέση (150, 73), στο 5ο frame του animation και κοιτάζει δεξιά και ο δεύτερος βρίσκεται στη θέση (412, 128), στο 3ο frame και κοιτάζει αριστερά. Όλες οι μεταβλητές που αναφέρθηκαν και για τους δύο παίχτες πρέπει να είναι γνωστές και στους δύο, για αυτό ο server και ο client ανταλλάζουν αυτά τα δεδομένα επαναληπτικά σε κάθε frame του παιχνιδιού με ένα πρωτόκολλο που περιγράφεται σε επόμενη ενότητα.

Ο server είναι υπεύθυνος για τη διατήρηση σωστών τιμών στις λίστες swords[] και slimes[]. Με άλλα λόγια, αναλαμβάνει να προσθέσει σπαθιά και εχθρούς στο παιχνίδι και παρακολουθεί πότε πρέπει να τα αφαιρέσει. Σε κάθε frame του παιχνιδιού αποφασίζει τυχαία αν θα προσθέσει ένα στοιχείο, με μία πολύ μικρή πιθανότητα προσθήκης.

```
491         # Take a chance at spawning enemies and swords.
492         if slimes_to_spawn > 0 and random() < 0.25 * delta_time:
493             spawn_slime(slimes)
494             slimes_to_spawn -= 1
495         if len(swords) < jo.MAX_SWORDS and random() < 0.07 * delta_time:
496             spawn_sword(swords)
```

Οι συναρτήσεις spawn_slime(slimes) και spawn_sword(swords) ενημερώνουν τις αντίστοιχες λίστες με κάποια τυχαιότητα ως προς το νέο στοιχείο.

```
    1 usage
47  def spawn_slime(all_slimes):
48      """ Add a new slime on the list of enemies. """
49      spawn_index = floor(random() * len(jo.levels[level_index].enemy_spawns))
50      new_slime = [jo.levels[level_index].enemy_spawns[spawn_index], 0, False]
51      all_slimes.append(new_slime)
52
53
    1 usage
54  def spawn_sword(all_swords):
55      """ Add a new sword on the list of active swords. """
56      new_index = floor(random() * jo.MAX_SWORDS)
57      while new_index in all_swords:
58          new_index = floor(random() * jo.MAX_SWORDS)
59      jo.ding_sound.play()
60      all_swords.append(new_index)
```

Για να αφαιρέσει τα στοιχεία, ο server ελέγχει αν υπάρχει collision του rectangle του κάθε χαρακτήρα με το rectangle κάθε σπαθιού ή εχθρού και ενημερώνει τις αντίστοιχες μεταβλητές. Το collision μεταξύ χαρακτήρα και εχθρού μπορεί να σημαίνει είτε απώλεια ζωής, είτε αφαίρεση του εχθρού από το παιχνίδι, ανάλογα με την κατάσταση του χαρακτήρα.

```
        # Check if there is conflict with an enemy.
        for slime_rect in slime_rects:
            if client_rect.colliderect(slime_rect):
                if client_anim_key == "attack" and client_can_kill:                 # Kill an enemy.
                    client_can_kill = False
                    enemies_killed += 1
                    slimes.pop(slime_rects.index(slime_rect))
                    slime_rects.remove(slime_rect)  # In case server and client kill the same enemy at the same frame.
                elif not client_anim_key == "attack" and not client_immune:         # Take damage.
                    client_immune = True
                    client_immune_frame = 0
                    jo.damage_sound.play()
                    hp -= 1
                break
        for slime_rect in slime_rects:
            if server_rect.colliderect(slime_rect):
                if attacking and server_can_kill:                   # Kill an enemy.
                    server_can_kill = False
                    jo.hit_kill_sound.play()
                    enemies_killed += 1
                    slimes.pop(slime_rects.index(slime_rect))
                elif not attacking and not server_immune:           # Take damage.
                    server_immune = True
                    server_immune_frame = 0
                    jo.damage_sound.play()
                    hp -= 1
                break
```

Όμως ο client πρέπει να γνωρίζει επίσης αν σκότωσε εχθρό, για να αναπαράγει τον αντίστοιχο ήχο και αν μάζεψε σπαθί, για να αυξήσει τις διαθέσιμες επιθέσεις του. Επομένως, και ο client υπολογίζει αυτά τα collisions, αλλά μόνο για τον δικό του χαρακτήρα. Η λογική για την αναπαραγωγή των ήχων είναι ότι η κάθε διεργασία αναπαράγει τους ήχους επιθέσεων και μαζέματος σπαθιών μόνο για τον δικό της χαρακτήρα, αφού αποτελούν προσωπικές ενέργειες, αλλά για την απώλεια ζωής και την εμφάνιση σπαθιών αναπαράγονται και στις δύο, γιατί αφορούν και τους δύο παίχτες.

Πληροφορίες που διαφέρουν για κάθε ένα από τα δύο επίπεδα αποθηκεύονται στη λίστα levels[], η οποία αποτελείται από δύο Level objects. Ο server και ο client διατηρούν μία μεταβλητή level_index για να γνωρίζουν σε ποιο επίπεδο παίζουν.

```
2 usages
class Level:
    def __init__(self, enemies_num, enemy_spawns, sword_spawns):
        self.enemies_num = enemies_num          # Number of enemies to spawn during gameplay. Killing all wins the level.
        self.enemy_spawns = enemy_spawns        # Positions of possible spawn points for slimes.
        self.sword_spawns = sword_spawns        # Positions of possible spawn points for extra attacks.
```

```
levels = [
    Level( enemies_num: 10,  enemy_spawns: [(36, 428), (1042, 129), (1764, 428)],
           sword_spawns: [(471, 172), (1250, 228), (1310, 709)]),
    Level( enemies_num: 15,  enemy_spawns: [(994, 134), (1648, 134), (1648, 907)],
           sword_spawns: [(52, 469), (1312, 168), (1189, 887)])
]
```

Αν η ομάδα καταφέρει να σκοτώσει και τους 10 εχθρούς του πρώτου επιπέδου, ο server ενημερώνει τον client ότι το gameplay πρέπει να διακοπεί. Ο client καταλαβαίνει ότι αφού οι ζωές δεν έχουν τελειώσει ακόμα, η διακοπή έγινε για αλλαγή επιπέδου. Η menu_screen γίνεται ‘4’ και στις δύο

διεργασίες και εμφανίζεται το ενδιάμεσο μενού ολοκλήρωσης επιπέδου. Εδώ γίνεται render το σκορ της ομάδας μέχρι τώρα, το οποίο ο server στέλνει στον client μόλις το υπολογίσει, και στην οθόνη του server υπάρχει επίσης ένα κουμπί Start. Όταν το Start πατηθεί με το ποντίκι, ξεκινάει αντίστροφη μέτρηση για την εκκίνηση του δεύτερου επιπέδου, ακριβώς όπως έγινε και στην οθόνη εκκίνησης. Στο δεύτερο επίπεδο, η menu_screen γίνεται '-1' πάλι και το gameplay ξαναρχίζει κανονικά, με μόνη διαφορά την τιμή του level_index, η οποία θα επηρεάσει την εικόνα της σκηνής, το κομμάτι της μουσικής υπόκρουσης, τις θέσεις των σπαθιών και εχθρών και το πλήθος των εχθρών που θα εμφανιστούν. Ανάμεσα στα δύο επίπεδα, διάφορες μεταβλητές, όπως οι θέσεις των παιχτών και το πλήθος των ζωών, αρχικοποιούνται πάλι.

Αν η ομάδα σκοτώσει και τους 15 εχθρούς του δεύτερου επιπέδου ή αν χάσει όλες της τις ζωές, σε οποιοδήποτε από τα δύο επίπεδα, ο server στέλνει σήμα για τη διακοπή του gameplay, η menu_screen γίνεται '5' και εμφανίζεται η τελική οθόνη με τον πίνακα τον τριών καλύτερων ομάδων και το τελικό σκορ της τρέχουσας ομάδας. Αυτή η οθόνη προσαρμόζεται ανάλογα με το αν πρόκειται για νίκη ή ήττα του παιχνιδιού. Το ίδιο ισχύει και για τον ήχο που θα αναπαραχθεί με το τέλος του παιχνιδιού. Οι παίχτες μπορούν να κλείσουν το παιχνίδι πατώντας Escape.

Κίνηση των NPC

Ο server αναλαμβάνει την κίνηση όλων των ζωντανών εχθρών σε κάθε frame του παιχνιδιού. Με την κλήση της συνάρτησης move_slimes(slimes) μέσα στο βασικό loop της pygame, η θέση, η φορά και ο δείκτης του animation κάθε εχθρού ενημερώνονται μέσα στη λίστα slimes[].

```
1 usage
def move_slimes(all_slimes):
    """ Track the movement and animation of all enemies for the current game frame. """
    offset_x = 20
    offset_y = 90
    inertia_x = 30
    for slime in all_slimes:
        # Update animation frame.
        slime[1] += jo.ENEMY_ANIM_STEP
        if slime[1] > jo.slime_num_frames["walk"] - 1:
            slime[1] = 0
        # Choose the player that is closer.
        server_pos = (server_x, server_y)
        client_pos = (client_x, client_y)
        if get_distance(slime[0], server_pos) <= get_distance(slime[0], client_pos):
            goal_pos = server_pos
        else:
            goal_pos = client_pos
        # Calculate the new position.
        if goal_pos[0] + (offset_x * jo.PLAYER_SCALE) < slime[0][0]:
            # When the distance is smaller than the step, use a smaller step to avoid passing the player.
            new_x = slime[0][0] - min(enemy_vel, abs(slime[0][0] - goal_pos[0]))
            # Introduce inertia in turning the other way.
            if abs(slime[0][0] - goal_pos[0]) + (offset_x * jo.PLAYER_SCALE) > inertia_x:
                slime[2] = False
        else:
            new_x = slime[0][0] + min(enemy_vel, abs(slime[0][0] - goal_pos[0]))
            if abs(slime[0][0] - goal_pos[0]) + (offset_x * jo.PLAYER_SCALE) > inertia_x:
                slime[2] = True
        if goal_pos[1] + (offset_y * jo.PLAYER_SCALE) < slime[0][1]:
            new_y = slime[0][1] - min(enemy_vel, abs(slime[0][1] - goal_pos[1]))
        else:
            new_y = slime[0][1] + min(enemy_vel, abs(slime[0][1] - goal_pos[1]))
        slime[0] = (new_x, new_y)
```

Η κάθε γλίτσα αποφασίζει ποιον χαρακτήρα να κυνηγήσει με βάση τη γεωμετρική της απόσταση από αυτούς.

```
2 usages
def get_distance(point1, point2):
    """ Calculate and return the distance between two given points. """
    return sqrt(((point2[0] - point1[0]) ** 2) + ((point2[1] - point1[1]) ** 2))
```

Αφού διαλέξει τον κοντινότερο παίχτη, αλλάζει τη θέση της με βάση μια ορισμένη ταχύτητα, η οποία είναι αισθητά χαμηλότερη από την ταχύτητα των παιχτών. Με τον υπολογισμό της φοράς της όμως, παρουσιάστηκε ένα πρόβλημα. Όταν η θέση μία γλίτσας στον οριζόντιο άξονα ήταν πολύ κοντά στον στόχο της, η φορά της άλλαζε σε διαδοχικά frames με αποτέλεσμα να μοιάζει σαν να αναβοσβήνει όσο κινούνταν κάθετα. Για την επίλυση αυτού του ζητήματος συνδυάστηκαν δύο παρεμβάσεις. Πρώτον, όταν η απόσταση που έχει να διανύσει από τον στόχο της είναι μικρότερη από αυτή που μπορεί να διανύσει, κινείται ακριβώς όσο χρειάζεται για να αγγίξει το επιθυμητό σημείο. Έτσι δεν το προσπερνάει και δε χρειάζεται να γυρίσει πίσω στο επόμενο frame. Δεύτερον, η φορά της δεν αλλάζει πάντα όταν ξεκινάει να κινείται ανάποδα, αλλά μόνο αν η απόσταση που έχει να διανύσει είναι μεγαλύτερη από 30 pixels. Αυτό δίνει την αίσθηση της αδράνειας στην αλλαγή της πλευράς που κοιτάζει, όταν κυνηγάει ένα παίχτη που αλλάζει κατεύθυνση.

Πρωτόκολλο επικοινωνίας

Πριν η ροή του παιχνιδιού να προχωρήσει από την αρχική οθόνη του μενού, ιδρύεται μία σύνδεση μεταξύ του server και του client στη θύρα 2000. Για μεμονωμένα μηνύματα μεταξύ server και client, η αποστολή ενός string μέσω του ανοιχτού socket είναι απλή. Αρκεί η μετατροπή του string σε bytes. Αυτό γίνεται με το όνομα της ομάδας, αλλά και το σήμα για την έναρξη της αντίστροφης μέτρησης και τελικά τη μετάβαση στο gameplay είναι απλά το string “start”. Η ανταλλαγή των δεδομένων του καθενός frame του παιχνιδιού κατά το gameplay όμως είναι πιο απαιτητική.

Οι δύο διεργασίες χρησιμοποιούν τις συναρτήσεις encode_frame_data(...) και decode_frame_data(frame_data) πριν στείλουν και αφού λάβουν τα δεδομένα του τρέχοντος frame. Όλες οι μεταβλητές που πρέπει να αποσταλούν γίνονται τιμές σε ένα dictionary, με κλειδιά τα ονόματά τους. Έτσι το καινούργιο dictionary είναι πλέον η μοναδική μεταβλητή που χρειάζεται να αποσταλεί. Με την built-in βιβλιοθήκη json της python, το dictionary μετατρέπεται σε μορφή JSON και τέλος, κωδικοποιείται σε bytes. Στην απέναντι πλευρά, η ίδια διαδικασία ακολουθείται ανάποδα και εξάγονται οι τιμές των αρχικών μεταβλητών. Παρ’ ότι οι ίδιες συναρτήσεις χρησιμοποιούνται και από τις δύο διεργασίες, ο server χρειάζεται να στείλει επιπλέον τις μεταβλητές που παρακολουθεί μόνο αυτός: τις ζωές, τα σπαθιά, τους εχθρούς και το σήμα τερματισμού του gameplay. Όταν ο client στέλνει δεδομένα, αυτά τα ορίσματα αγνοούνται.

Για τη διαχείριση σφαλμάτων που μπορεί να προκύψουν σε αυτό το στάδιο της επικοινωνίας, εκτός από την εξαίρεση socket.error που ελέγχεται σε κάθε αποστολή και λήψη δεδομένων, λαμβάνεται υπ’ όψη και η εξαίρεση json.decoder.JSONDecodeError η οποία μπορεί να προκύψει από τη λήψη κενού μηνύματος.

```
2 usages
def encode_frame_data(anim_key, anim_index, flipped, pos, attacks, hp=None, slimes=None, swords=None, stop=False):
    """ Compose and encode the data for the current frame to send to the teammate, using a custom protocol.

    Parameters:
        anim_key (string): Dictionary key for the list of the animation used by the player.
        anim_index (int): Frame index for the animation list used by the player.
        flipped (boolean): Whether the rendered frame was flipped horizontally.
        pos (tuple of floats): The position of the frame on the screen.
        attacks (int): The number of available hits left.
        hp (int): Health points of the team.
        slimes (list of tuples): Data for each enemy currently in-game. Could be empty.
        swords (list of ints): Indexes of sword spawns of swords currently in-game. Could be empty.
        stop (boolean): Whether the gameplay must stop and the screen mode to change.

    Returns:
        (bytes): Data ready to be sent through the custom protocol.
    """

    var_dict = {
        "anim_key": anim_key,
        "anim_index": anim_index,
        "flipped": flipped,
        "pos": pos,
        "attacks": attacks,
        "hp": hp,
        "slimes": slimes,
        "swords": swords,
        "stop": stop
    }
    json_data = json.dumps(var_dict)
    return json_data.encode()
```

```
2 usages
def decode_frame_data(frame_data):
    """ Decode and parse the data received from the teammate for the current frame, using a custom protocol.

    Parameters:
        frame_data (bytes): Data received through the custom protocol.

    Returns:
        tuple of:
            anim_key (string): Dictionary key for the list of the animation used by the player.
            anim_index (int): Frame index for the animation list used by the player.
            flipped (boolean): Whether the rendered frame was flipped horizontally.
            pos (tuple of floats): The position of the frame on the screen.
            attacks (int): The number of available hits left.
            hp (int): Health points of the team.
            slimes (list of tuples): Data for each enemy currently in-game. Could be empty.
            swords (list of ints): Indexes of sword spawns of swords currently in-game. Could be empty.
            stop (boolean): Whether the gameplay must stop and the screen mode to change.
    """

    json_data = frame_data.decode()
    var_dict = json.loads(json_data)
    anim_key = var_dict["anim_key"]
    anim_index = var_dict["anim_index"]
    flipped = var_dict["flipped"]
    pos = var_dict["pos"]
    attacks = var_dict["attacks"]
    hp = var_dict["hp"]
    slimes = var_dict["slimes"]
    swords = var_dict["swords"]
    stop = var_dict["stop"]
    return anim_key, anim_index, flipped, pos, attacks, hp, slimes, swords, stop
```

Στο σημείο του pygame loop όπου καλούνται αυτές οι συναρτήσεις, η σειρά τους είναι ανάποδα στον server από ότι στον client. Πρώτος στέλνει τα δεδομένα του ο server και ο client ακούει και μετά στέλνει ο client τα δικά του και ο server τα λαμβάνει. Με παρόμοια λογική αποστέλλονται τα δεδομένα της βάσης δεδομένων από τον server στον client στο τέλος του παιχνιδιού.

Βάση δεδομένων & leaderboard

Ένα ακόμη χρέος που βαραίνει τον server είναι η διατήρηση μία μικρής βάσης δεδομένων sqlite με τα ονόματα και τα καλύτερα σκορ κάθε ομάδας που έχει παίξει σε αυτή τη συσκευή. Η βάση αποτελείται από έναν μόνο πίνακα με 3 στήλες: το id κάθε καταχώρησης, το όνομα της ομάδας και το καλύτερο μέχρι τώρα σκορ αυτής της ομάδας.

	id	name	score
	Search column...	Search column...	Search column...
1	1	the skeletons	180
2	2	zombie fam	0
3	3	sax and piano	10
4	4	gamingTeam	200
5	5	game zombies	240
6	6	new team	30
7	7	testing	220
8	8	sixth team	10
9	9	chicken dinner	270
10	10	Buena Vista Jazz Club	80
11	11	we are Legend	290
12	12	WE ARE legend	20
13	13	gaming dead	240
14	14	Night of the dev	280
15	15	DAWN OF THE DEV	390
16	16	Bones and Game engines	200

Κάθε ομάδα ξεχωρίζει από το όνομα που εισάγει στην αρχή του παιχνιδιού. Αν οι ίδιοι παίχτες χρησιμοποιήσουν διαφορετικό όνομα θα θεωρηθούν νέα ομάδα. Αν χρησιμοποιήσουν ένα όνομα που υπάρχει ήδη, ίσως καταφέρουν να αλλάξουν το high score αυτής της ομάδας.

Στο τέλος του παιχνιδιού, είτε πρόκειται για νίκη είτε όχι, ο server θα ψάξει το όνομα της ομάδας στη βάση. Αν δεν υπάρχει, θα προσθέσει μία καταχώρηση με το σκορ της παρτίδας που μόλις τελείωσε. Αν υπάρχει, θα συγκρίνει το τωρινό σκορ με αυτό που είναι καταχωρημένο και αν το καινούργιο είναι υψηλότερο, θα ενημερώσει τη βάση. Μετά θα ψάξει τις 3 καλύτερες ομάδες και θα κρατήσει τα ονόματά τους και τα σκορ τους. Τέλος, θα υπολογίσει ποια είναι η κατάταξη της τωρινής παρτίδας στον πίνακα τον ομάδων, μετρώντας όλες τις γραμμές με υψηλότερο ή ίσο σκορ. Οι ισοπαλίες κρίνονται υπέρ της παλαιότερης καταχώρησης (ούτως ή άλλως τα επιτεύγματά σου δοξάζονται περισσότερο όταν είσαι ο πρώτος που τα πέτυχε).

```
454             # Access the database and update it, if appropriate.
455             sql_select_team = """
456                 SELECT * FROM teams WHERE name = '""" + team_name + """';
457             """
458             team_entry = execute_read_query(db_conn, sql_select_team)
459             if len(team_entry) == 0:
460                 sql_insert_team = """
461                     INSERT INTO teams (name, score)
462                         values
463                         ('""" + team_name + """', """ + str(final_score) + """);
464                 """
465                 execute_write_query(db_conn, sql_insert_team)
466             elif team_entry[0][2] < final_score:
467                 sql_update_team = """
468                     UPDATE teams SET score = """ + str(final_score) + """ WHERE name = '""" + team_name + """';
469                 """
470                 execute_write_query(db_conn, sql_update_team)
471             sql_select_top = """
472                 SELECT name, score FROM teams ORDER BY score DESC LIMIT 3;
473             """
474             top_teams = execute_read_query(db_conn, sql_select_top)
475             sql_calculate_rank = """
476                 SELECT COUNT(*) FROM teams WHERE score >= """ + str(final_score) + """;
477             """
478             rank_result = execute_read_query(db_conn, sql_calculate_rank)
479             team_rank = rank_result[0][0]       # Rank for this run's score, not the best score of the team.
480             # Send data derived from the database to the client.
481             db_data = jo.encode_db_data(top_teams, team_rank)
482             try:
483                 conn.sendall(db_data)
484             except socket.error:
485                 print("Failed to send database data to client.")
```

Η θέση στην κατάταξη είναι το πλήθος των ομάδων που μετρήθηκαν προσαυξημένο κατά 1, όμως έχοντας ήδη προσθέσει αυτή την ομάδα στη βάση, έχει μετρηθεί και ο εαυτός της. Άρα η ανάθεση

$$team\_rank = rank\_result[0][0] + 1 - 1$$

θα απλουστευτεί σε

$$team\_rank = rank\_result[0][0]$$

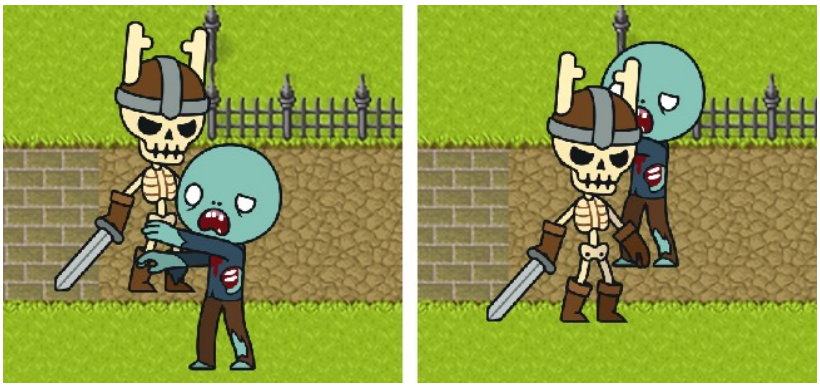
Οι ευρετηριάσεις των rank_result και team_entry είναι απαραίτητες, γιατί η βάση επιστρέφει μία λίστα από tuples. Αν μία ομάδα έχει υψηλότερο σκορ καταχωρημένο στη βάση από της τωρινής παρτίδας, η κατάταξή της υπολογίζεται για την τωρινή παρτίδα, για να αξιολογήσει την επίδοση αυτού του παιχνιδιού.

Έχοντας πλέον όλες τις πληροφορίες που χρειάζονται για το rendering της τελική οθόνης, ο server στέλνει τις κορυφαίες 3 ομάδες και τον αριθμό της κατάταξης στον client. Η αποστολή του σκορ της ομάδας έχει προηγηθεί της διαχείρισης της βάσης δεδομένων. Για την αποστολή των top_teams και team_rank χρησιμοποιείται η συνάρτηση encode_db_data(top_teams, team_rank) από τον server και η decode_db_data(db_data) από τον client. Αυτές οι συναρτήσεις ακολουθούν τη διαδικασία μετατροπής σε dictionary και μετά σε JSON που περιγράφτηκε στην ενότητα του πρωτοκόλλου επικοινωνίας.

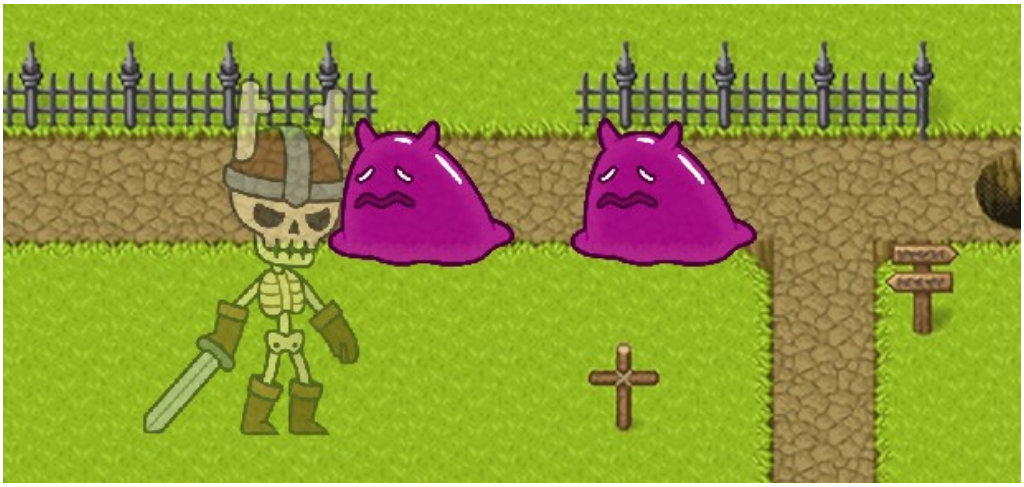
Ειδικές περιπτώσεις & κολπάκια

Η κίνηση των χαρακτήρων υπολογίζεται ξεχωριστά για τον κάθε άξονα, όμως η αναμενόμενη συμπεριφορά είναι μία σταθερή ταχύτητα ασχέτως κατεύθυνσης. Όταν ένας παίχτης κινείται διαγώνια, θα πρέπει να διανύει την ίδια απόσταση που θα διένυε με οριζόντια ή κατακόρυφη κίνηση στον ίδιο χρόνο. Επομένως, η πρόσθεση ενός βήματος στη θέση δεν μπορεί να είναι ανεξάρτητη στους δύο άξονες. Έστω simple_vel το βήμα του χαρακτήρα όταν κινείται μόνο πάνω/κάτω ή μόνο δεξιά/αριστερά. Από το Πυθαγόρειο θεώρημα, το βήμα του για διαγώνια κίνηση θα είναι diagonal_vel = sqrt(simple_vel * simple_vel / 2).

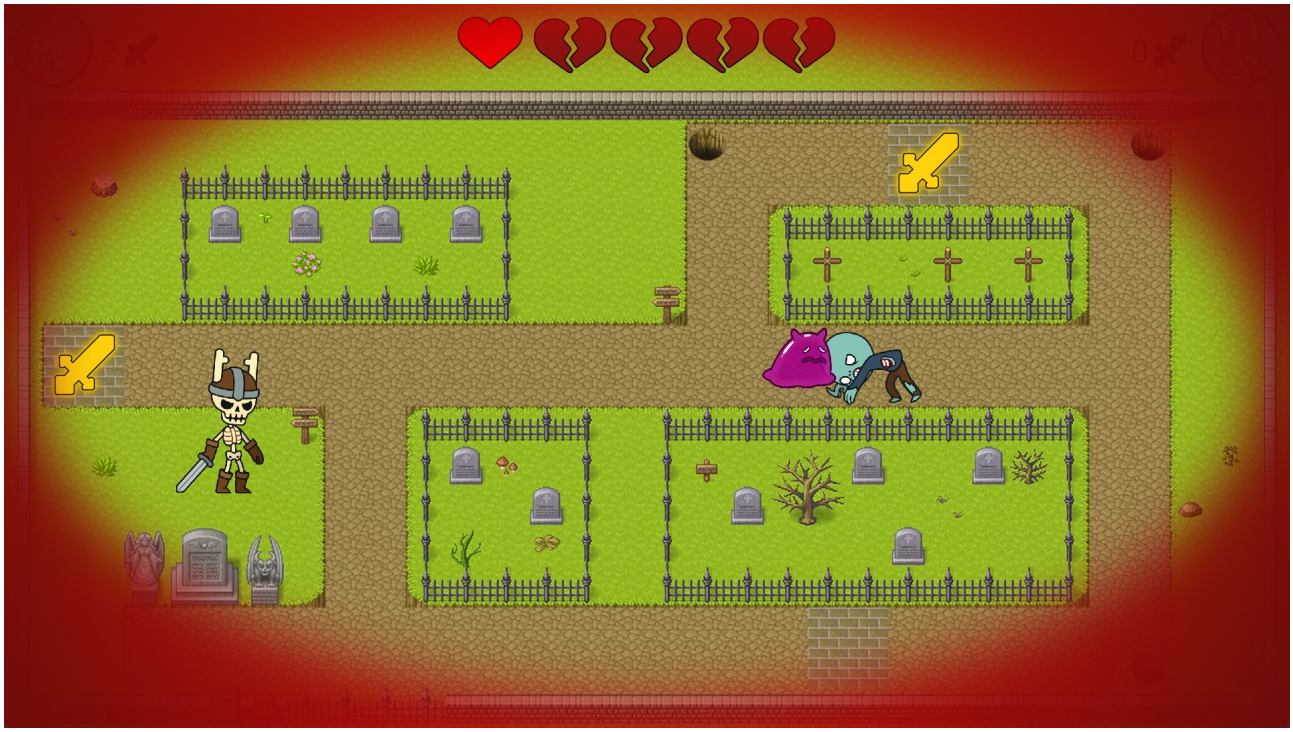
Σε ένα 2D top down παιχνίδι, η κατανόηση του βάθους από τον παίχτη εξαρτάται από διάφορους παράγοντες. Ένας από αυτούς είναι η σειρά με την οποία γίνονται render τα αντικείμενα στην οθόνη. Αντικείμενα που είναι πιο χαμηλά πρέπει να δίνουν την εντύπωση ότι είναι πιο μπροστά. Για αυτό η σειρά που οι δύο χαρακτήρες γίνονται blit στην οθόνη εξαρτάται από τη θέση τους στον κάθετο άξονα.



Σε παιχνίδια αυτού του genre, όταν ένας παίχτης είναι περικυκλωμένος και πρόκειται να χάσει ζωή, συνηθίζεται να χάνει μόνο μία και να έχει την ευκαιρία να αλλάξει θέση για να συνεχίσει. Για να επιτευχθεί αυτό στο «*Οι νεκροί ακούνε τζαζ*!», μετά από απώλεια ζωής, ακολουθεί μία περίοδος ανοσίας 2,4 δευτερολέπτων. Η ύπαρξη της ανοσίας έχει και πρακτική αξία, καθώς χωρίς αυτήν, collisions μεταξύ εχθρού και χαρακτήρα σε διαδοχικά frames θα αφαιρούσαν όλα ζωές, δίνοντας την αίσθηση ότι ένας εχθρός κατανάλωσε και τις 5 ζωές. Η διάρκεια της ανοσίας μεταδίδεται στον παίχτη κάνοντας τον χαρακτήρα προσωρινά ημιδιάφανο. Για να γίνει ημιδιαφανές το surface του χαρακτήρα, πολλαπλασιάζεται με ένα surface με RGBA τιμή (255, 255, 255, 128) και μετά η τιμή κάθε pixel διαιρείται με 256, χάρη στο special flag pygame.BLEND_RGBA_MULT.



Ένα επιπρόσθετο εφέ που χρησιμοποιείται είναι αυτό της τελευταίας ζωής. Όταν η ομάδα έχει μόνο μία ζωή, το πλαίσιο της οθόνης κοκκινίζει, κάνοντας render μία εικόνα με διαφάνεια πάνω από τις υπόλοιπες στρώσεις.



Δυσκολίες & προκλήσεις

Η διαχείριση της αποστολής δεδομένων σε κάθε frame του παιχνιδιού ήταν μία μικρή πρόκληση. Γνωρίζοντας πλέον πόσο προβληματική μπορεί να γίνει η ανταλλαγή δεδομένων μεταξύ server και client, χωρίς να δοθεί άδικο πλεονέκτημα στον παίχτη που παίζει από την πλευρά του server, αλλά και χωρίς να γίνει δυσανάγνωστος ο κώδικας μέσα στο βασικό pygame loop, στο μέλλον ίσως να προτιμούσα μία αρχιτεκτονική με 2 πανομοιότυπους clients και έναν ανεξάρτητο server. Αν και απέφυγα αυτή την προσέγγιση, για να μην αυξήσω την πολυπλοκότητα, ίσως η καθυστέρηση της επικοινωνίας να διατηρούνταν δίκαιη για τους δύο παίχτες ευκολότερα.

Ένα άλλο θέμα που προέκυψε ήταν το screen tearing. Η pygame έχει μία ρύθμιση για να αρχικοποιηθεί με χρήση vsync, αλλά σύμφωνα με το documentation, δεν υπάρχει εγγύηση ότι θα λειτουργήσει. Στον δικό μου υπολογιστή, η προσθήκη αυτού του flag στην pygame.display.set_mode() δεν είχε καμία διαφορά.

Τέλος, το balancing ενός local-multiplayer παιχνιδιού από ένα άτομο ήταν δυσλειτουργικό. Ευτυχώς, το testing ήταν εύκολο, έχοντας και τις δύο διεργασίες ανοιχτές στο ίδιο μηχάνημα, οπότε δε τίθεται θέμα debugging. Όμως, δεν είμαι σίγουρος ότι το παιχνίδι δεν είναι πιο εύκολο ή πιο δύσκολο από όσο θα έπρεπε, αφού δεν έχω σχεδόν καθόλου δεδομένα πραγματικής χρήσης. Μελλοντικές αλλαγές θα μπορούσαν να γίνουν σε διάφορα μεγέθη του παιχνιδιού, μέσω σταθερών στο αρχείο *jazz_operations.py*.

Συζήτηση

Αυτή η εργασία ήταν η πρώτη μου επαφή με το multiplayer game development και μου έμαθε πολλά για την επικοινωνία μεταξύ δύο υπολογιστών στα πλαίσια μίας εφαρμογής πραγματικού χρόνου. Η ανταλλαγή δεδομένων ανά καρέ είναι πιο απαιτητική από οποιαδήποτε ασύγχρονη δικτυακή επικοινωνία και αναγκάζει τον developer να δώσει προτεραιότητα στο performance. Ο χρόνος αποστολής δεδομένων αθροίζεται με τον χρόνο των υπόλοιπων υπολογισμών, επομένως, αν στόχος είναι τα 60 FPS σε server και client, πρέπει ολόκληρη η διαδικασία να ολοκληρωθεί σε λιγότερο από 16ms. Σε LAN χρήση φυσικά αυτό είναι πολύ πιο εύκολο από ότι μέσω internet, με τον κάθε παίχτη να βρίσκεται σε άλλη χώρα, όπως θα γινόταν σε ένα ρεαλιστικό σενάριο multiplayer παιχνιδιού. Έχοντας ολοκληρώσει την εργασία, καταλαβαίνω ότι αν είχα χρησιμοποιήσει νήματα για να απομονώσω την επικοινωνία server και client από το rendering του κάθε frame, θα είχα μεγαλύτερη ανεξαρτησία των frame rates και ανοχή σε απώλεια πακέτων. Από την άλλη πλευρά, η εισαγωγή πολυνηματικού προγραμματισμού στην εργασία θα είχε δημιουργήσει άλλες δυσκολίες που δε θα προλάβαινα να επιλύσω έγκαιρα.

Η προσέγγιση της pygame, με χειροκίνητη διαχείριση του rendering κάθε frame, ήταν χαμηλότερης αφαιρετικότητας από ότι είχα συνηθίσει και σίγουρα πολύ χρήσιμη για να αποκτήσω σφαιρικότερες γνώσεις στον προγραμματισμό παιχνιδιών. Παρ’ όλα αυτά, η εργασία της Αλγοριθμικής Σκέψης ήταν και η πιο απαιτητική του εξαμήνου και ήταν λίγο απογοητευτικό που χρειάστηκε να αφιερώσω τόσον χρόνο στην εξοικείωση με μία τεχνολογία που δεν πρόκειται να χρησιμοποιήσω ποτέ εκτός ακαδημαϊκού περιβάλλοντος. Αν και ο προγραμματισμός εκτός game engine είναι θεμιτός, θα προτιμούσα η χρονοβόρα εργασία του εξαμήνου να ήταν σε κάποια τεχνολογία που χρησιμοποιείται από τη βιομηχανία.

Λοιποί συντελεστές

Τα sprites του σκελετού και του ζόμπι δημιουργήθηκαν από την Irina Mir (irmirx) και έχουν άδεια CC BY 3.0. Το sprite της γλίτσας δημιουργήθηκε από τον Penzilla με άδεια χρήσης που απαγορεύει μόνο το redistribution. Το πλαίσιο του παραθύρου και του κουμπιού στο μενού, καθώς και τα εικονίδια καρδιών και σπαθιού προέρχονται από το Kenney, με άδεια CC0. Η εικόνα με το μοτίβο των κρανίων στην οποία βασίστηκε το background του μενού δημιουργήθηκε από τον Tamalee (drabbitod) και έχει άδεια χρήσης που απαγορεύει μόνο το redistribution. Οι εικόνες που χρησιμοποιήθηκαν ως σκηνές για τα δύο επίπεδα δημιουργήθηκαν με το open-source εργαλείο TileMap Editor του Todor Imreorov (blurymind). Όσα γραφικά δεν αναφέρονται εδώ, τα έφτιαξα εγώ, είτε μόνος μου, είτε με επεξεργασία των παραπάνω στο GIMP.

Το animation της γλίτσας είχε μόνο 4 frames, τα οποία δεν αρκούσαν για αναπαραγωγή στην επιθυμητή ταχύτητα. Κάνοντας frame interpolation με το open-source εργαλείο Flowframes του N00MKRAD, το οποίο χρησιμοποιεί AI, δημιουργήθηκαν τα 12 animation frames που περιέχονται στο project.

Η γραμματοσειρά του UI είναι η Aka-Acid Dosis που σχεδιάστηκε από την Μυρτώ Ορφανουδάκη, με άδεια χρήσης SIL Open Font License.

Η μουσική του παιχνιδιού βρέθηκε μέσω του Pixabay και έχει άδεια που απαγορεύει μόνο το redistribution. Το κομμάτι που παίζει στο μενού είναι το “the best jazz club in New Orleans” του Paolo Argento. Τα κομμάτια των δύο επιπέδων είναι τα “upbeat mission fun and quirky adventure” του Cyberwave Orchestra και “in jazz” του Vlad Krotov, αντίστοιχα.

Οι ήχοι του παιχνιδιού δημιουργήθηκαν με foley που βρέθηκαν μέσω του Freesound. Κάποιοι χρησιμοποιήθηκαν αυτούσιοι, ενώ άλλοι χρειάστηκαν επεξεργασία στο Audacity. Τα original αρχεία και οι άδειες χρήσεις τους φαίνονται στον παρακάτω πίνακα.

Ήχος	**Δημιουργός**	**Άδεια Χρήσης**
SlimeJump	Zuzek06	CC0
Basic Melee Swing / Miss / Whoosh	SypherZent	CC0
Echoing Glass Bell Ringing	f-r-a-g-i-l-e	CC BY 4.0
06SWORD06	lostchocolatelab	CC0
AltoSax1	fredgalist	CC BY 3.0
Metal Spawn_3	eardeer	CC BY 4.0
20IstanbulMehmetRideCymbalEdge	pjcohen	CC BY 4.0