PENERAPAN *DYNAMIC AUDIO* PADA *2D ENDLESS RUNNER GAME* MENGGUNAKAN *PURE DATA*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Ade Darmawan
NIM: 155150200111106

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

PENGESAHAN

PENERAPAN *DYNAMIC AUDIO* PADA *2D ENDLESS RUNNER GAME* MENGGUNAKAN *PURE DATA*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Ade Darmawan
NIM: 155150200111106

Skripsi ini telah diuji dan dinyatakan lulus pada
18 Februari 2019
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Eriq Muhammad Adams Jonemaro, S.T, M.Kom	Tri Afirianto, S.T, M.T
NIP: 19850410 201212 1 001	NIK: 201309 851213 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar referensi.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 18 Februari 2019

Ade Darmawan

NIM: 155150200111106

PRAKATA

Puji dan syukur penulis penjatkan kepada Allah Subhanahu Wa Ta'ala atas segala rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "Penerapan *Dynamic Audio* Pada *2D Endless Runner Game* Menggunakan *Pure Data*" sebagai salah satu persyaratan untuk menyelesaikan studi di Jurusan Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya dengan baik dan tepat waktu.

Dalam kesempatan ini saya selaku penulis skripsi ini menyampaikan rasa terima kasih sebesar-besarnya kepada :

1. Keluarga yaitu kedua orang tua saya (bapak dan ibu) beserta adik saya yang selalu memberi dukungan serta doa.
2. Bapak Eriq Muhammad Adams Jonemaro, S.T, M.Kom selaku pembimbing I dan Tri Afirianto, S.T, M.T selaku pembimbing II yang telah membimbing dan mengarahkan penulis sehingga dapat menyelesaikan skripsi ini.
3. Seluruh dosen Fakultas Ilmu Komputer Universitas Brawijaya atas kesediaan membagi ilmunya kepada penulis sehingga penulis dapat menyelesaikan skripsi ini.
4. Kanjeng Mami dan Papa Jhon yang telah menyediakan tempat dan dukungan saat mengerjakan skripsi.
5. ANK yang telah memberi semangat hingga skripsi ini bisa selesai.
6. Teman-teman Slathem Studio yang selalu memotivasi dan memberi semangat kepada penulis di dalam penyelesaian skripsi ini.
7. Teman-teman seperjuangan Fakultas Ilmu Komputer yang telah memberikan bantuan selama masa studi hingga penyelesaian skripsi ini.
8. Pesenkopi dan D'Keprek yang telah memberi asupan gizi hingga skipsi ini bisa diselesaikan.

Penulis sadar bahwa dalam penyusunan skripsi ini masih banyak kekurangan, sehingga saran dan kritik yang membangun dapat disampaikan secara langsung untuk pengembangan dan penelitian selanjutnya.

Malang, 18 Februari 2019

Penulis
darmawan.ade98@student.ub.ac.id

ABSTRAK

Ade Darmawan, Penerapan *Dynamic Audio* Pada *2D Endless Runner Game* Menggunakan *Pure Data*

Pembimbing: Eriq Muhammad Adams Jonemaro, S.T, M.Kom dan Tri Afirianto, S.T, M.T

2D platformer game adalah *genre game* yang di mana pemain menggerakkan karakter melalui sisi samping kamera dan diharuskan melewati rintangan yang ada untuk mencapai objektif tertentu. Teknik dalam pengembangan *game* untuk menambah pengalaman bermain juga berkembang sangat pesat. Dapat diambil contoh yaitu penerapan *dynamic audio*. Penggunaan *dynamic audio* dalam *game* akan memberikan kelebihan daripada menggunakan *static audio* dengan memasukkan beberapa *asset* untuk tiap gerakan. Penelitian ini menggunakan *Pure Data* untuk mengimplementasikan *dynamic audio* pada *2D endless runner game*. *Pure Data* memiliki cara *visual* untuk memprogram suara, yaitu dengan menghubungkan kotak-kotak kecil bersama dengan kabel *virtual*. Oleh karena itu penggunaan *Pure Data* untuk menerapkan *dynamic audio* pada *2D endless runner game* sangat disarankan dengan mempertimbangkan kemudahan dalam menerapkannya. Hasil dari penelitian ini menunjukkan bahwa dari rancangan yang telah ditentukan sebelumnya didapatkan hasil valid dan berhasil dijalankan pada *2D endless runner game* yang memiliki FPS tidak berbeda signifikan dengan penggunaan *static audio.* Rata-rata FPS yang didapat yaitu 62,4 FPS untuk dynamic audio dan 62,1 FPS untuk *static audio* yang dapat disimpulkan bahwa penerapan *dynamic audio* pada *2D Endless Runner Game* tidak memberikan perbedaan performa.

Kata kunci: *2D Platformer, Endless Runner Game, Dynamic Audio, Pure Data*

ABSTRACT

Ade Darmawan, ***Application Of Dynamic Audio In Endless Runner 2D Games Using Pure Data***

Supervisors: **Eriq Muhammad Adams Jonemaro, S.T, M.Kom dan Tri Afirianto, S.T, M.T**

2D platformer game is a genre game where players move characters through a side camera and are required to pass existing obstacles to achieve certain goals. Techniques in game development to add to the playing experience are also growing very rapidly. Can be taken an example of the application of dynamic audio. The use of dynamic audio in the game will provide the advantage of using static audio by entering multiple assets for each movement. This research uses Pure Data to implement dynamic audio in endless 2D game runners. Pure Data has a visual way to program sound, namely by connecting small boxes together with virtual cables. Therefore the use of Pure Data is to apply dynamic audio to endless 2D game runners that are highly preferred with consideration of convenience in applying them. The results of this study prove that from predetermined designs obtained valid results and successfully run on 2D runner endless games that have FPS are not significantly different from the use of static audio. The average FPS obtained is 62.4 FPS for dynamic audio and 62.1 FPS for static audio which can be ignored as well as the application of dynamic audio in 2D Endless Runner Games does not provide performance.

Keywords*: 2D Platformer, Endless Runner Game, Dynamic Audio, Pure Data*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

Perkembangan dalam dunia *game* telah berkembang dengan sangat pesat (Iftikhar, 2015). Seiring perkembangan tersebut, muncul banyak *genre game* salah satunya *platformer game*. *Platformer game* dapat dipecah lagi menjadi dua tipe yaitu *platformer puzzle* dan *action* (Bhosale, 2018). Dapat diambil contoh *game Limbo* sebagai *game platformer puzzle* dan *Super Mario Bros* sebagai *game platformer action*. *Super Mario Bros* adalah salah satu *game platformer* populer yang diterbitkan oleh *Nintendo*. Setelah rilisnya *Super Mario Bros* ke pasaran, banyak *game platformer* lain yang bermunculan khususnya *2D platformer* seperti *Sonic The Hedgehog*. *2D platformer game* termasuk *game side scrolling* yang yang berarti *game* tersebut dilihat dari samping sisi sudut kamera. Kemudian muncul *2D platformer* bertipe *endless runner* yang berarti *game* ini tak memilik akhir dan menuntut pemain untuk mendapatkan skor setinggi tingginya. *2D endless runner* banyak diminati karena kesederhanaan cara bermainnya terutama memudahkan pemain yang memainkannya pada perangkat *mobile* (Jos, 2014).

Teknik dalam pengembangan *game* untuk menambah pengalaman bermain juga berkembang sangat pesat (Iftikhar, 2015). Dapat diambil contoh yaitu penerapan *dynamic audio*. *Dynamic audio* merupakan *audio* interaktif dalam *game* yang dapat langsung merespon masukan dari pemain dan membalasnya dengan *audio* yang dibutuhkan. Penggunaan *dynamic audio* dalam *game* akan memberikan kelebihan daripada menggunakan *static audio* dengan memasukkan beberapa *asset* untuk tiap gerakan. Kelebihan yang didapat yaitu tidak perlu memasukkan terlalu banyak *asset* ke dalam *game* untuk beberapa efek suara, yang berarti *game* dapat dibuat dengan ukuran yang lebih kecil daripada menggunakan *static audio*. Kelebihan yang lain adalah mudahnya mengatur variasi suara ketika menggunakan *dynamic audio* dengan cara menaikkan atau menurunkan frekuensi, tempo, dan komponen suara yang lain (Peerdeman, 2006). Hal itu berujung pada sedikitnya *asset* untuk variasi suara yang diperlukan yang berarti kita tidak perlu membuat *asset* suara baru jika menginginkan efek suara baru. Salah satu contoh penerapan *dynamic audio* yaitu saat melakukan gerakan meloncat pada karakter yang diam dan gerakan meloncat pada karakter yang berlari maka suara yang dikeluarkan akan beda atau suara langkah kaki pada saat karakter berlari dan berjalan, karena dalam praktiknya di dunia nyata tentu saja suara yang dihasilkan akan berbeda. *Dynamic audio* akan memberikan pengalaman bermain lebih natural dan imersif. Cara kerja *dynamic audio* sendiri adalah dengan mengolah suara langsung secara *digital*. Seperti contohnya menaikkan atau menurunkan frekuensi dari gelombang suara dari *asset* yang ada.

Penelitian ini menggunakan *Pure Data* untuk mengimplementasikan *dynamic audio* pada *2D endless runner game*. *Pure Data* adalah bahasa *visual programming* untuk multimedia yang dikembangkan oleh Miller Puckette. Pemrograman dalam *Pure Data* menyerupai merancang dan membangun mesin

untuk melakukan tugas yang diinginkan (Hancock, 2014). Penggunaan *Pure Data* sendiri dipilih agar mempermudah mengimplementasikan *dynamic audio* pada *2D endless runner game* dan kompatibilitasnya yang hampir mendukung semua *platform*.

Penelitian ini akan membahas tentang menerapkan *dynamic audio* pada *2D endless runner game* menggunakan *Pure Data* dan *Heavy compiler*. *Heavy* merupakan *compiler* yang dapat mengubah *module Pure Data* menjadi kode *C# native* yang dapat langsung diimplementasikan pada *Unity3D*. Hasil yang diharapkan dari penelitian ini yaitu menunjukkan kelebihan penggunaan *dynamic audio* khususnya menggunakan *Pure Data* dibanding penggunaan *static audio* pada *2D endless runner game*.

1.2 Rumusan masalah

Dari latar belakang di atas, didapatkan dua permasalahan yaitu :

1. Bagaimana cara membuat *audio game* yang dinamis pada *2D endless runner game*?
2. Bagaimana kinerja *dynamic audio* menggunakan *Pure Data* pada *2D endless runner game*?

1.3 Tujuan

Terwujudnya pengimplementasian *dynamic audio* menggunakan *Pure Data* pada *2D endless runner game*.

1.4 Manfaat

Penelitian ini diharapkan dapat memberikan manfaat antara lain :

a. Bagi pengguna :
    1. Sebagai sarana hiburan dan merasakan pengalaman bermain *game* yang menerapkan *dynamic audio*.
    2. Sebagai metode pembelajaran tentang *dynamic audio* menggunakan *Pure Data*.

b. Bagi penulis :

    Untuk memenuhi syarat kelulusan S1 di Universitas Brawijaya.

c. Bagi dosen :

    Sebagai sarana publikasi atau penyebaran jurnal ilmiah yang telah ditulis dan siap untuk disampaikan ke masyarakat.

1.5 Batasan masalah

Untuk membatasi ruang lingkup penelitian maka dibuat batasan masalah supaya lebih mudah dan fokus dalam perancangan dan implementasi. Batasan masalah pada penelitian ini adalah sebagai berikut:

1. *Platform* yang digunakan adalah *PC* dengan sistem operasi *Windows*.
2. *Game* berjenis *2D endless runner*.
3. Implementasi menggunakan *Unity3D*, *Pure Data*, dan *Heavy*.
4. Penelitian ini tidak membahas tentang pembuatan *game*.

1.6 Sistematika pembahasan

Sistematika penulisan yang disusun pada tugas akhir ini diharapkan dapat menunjang tujuan yang diharapkan. Adapun sistematika pada penulisan tugas akhir ini, sebagai berikut :

BAB I PENDAHULUAN

Bab ini membahas tentang latar belakang penulisan penelitian, rumusan masalah, tujuan penelitian, manfaat penelitian, sistematika penulisan yang digunakan untuk penerapan *dynamic audio* menggunakan *Pure Data* pada *2D endless runner game*.

BAB II LANDASAN KEPUSTAKAAN

Bab ini membahas tentang landasan kepustakaan yang relevan dengan penelitian.

BAB III METODOLOGI

Bab ini membahas tentang langkah-langkah yang digunakan untuk penerapan *dynamic audio* menggunakan *Pure Data* pada *2D endless runner game* dan langkah kerja yang dilakukan dalam penelitian.

BAB IV PERANCANGAN SISTEM

Bab ini membahas tentang rancangan penerapan *dynamic audio* menggunakan *Pure Data* pada *2D endless runner game*.

BAB V IMPLEMENTASI

Bab ini membahas tentang implementasi *dynamic audio* menggunakan *Pure Data* pada *2D Endless Runner Game.*

BAB VI PENGUJIAN DAN ANALISIS

Bab ini membahas tentang pengujian serta analisis terhadap penerapan *dynamic audio* menggunakan *Pure Data* pada *2D endless runner game*.

BAB VII PENUTUP

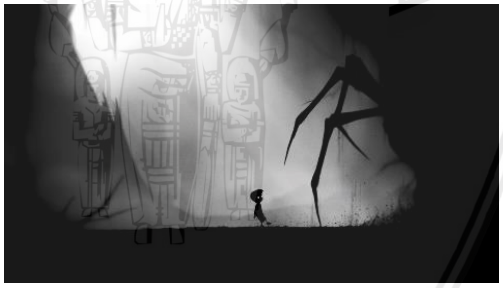
Bab ini berisi tentang kesimpulan yang diperoleh dari pembuatan dan pengujian peneperapan *dynamic audio* menggunakan *Pure Data* pada *2D endless runner game* yang dikembangakan dalam skripsi ini serta saran-saran untuk pengembangan sistem selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 *2D Platformer Game*

2D platformer game adalah *genre game* yang di mana pemain menggerakkan karakter melalui sisi samping kamera dan diharuskan melewati rintangan yang ada untuk mencapai objektif tertentu (Bhosale, 2018). *2D platformer game* itu sendiri tidak dapat sepenuhnya dianggap sebagai *genre standalone* karena sering dikombinasikan dengan *genre* permainan lain seperti *endless runner*, *RPG*, dan sebagainya. *Genre* ini berasal dari *arcade* di awal tahun 1980 yang memperkenalkan *game* seperti *Space Panic* dan *Donkey Kong* (Gustafsson, 2014).

Game ini biasanya mengharuskan pemain untuk mendapatkan *item* yang berceceran pada *level*, mengalahkan musuh yang ada, menghindari rintangan sampai melawan *boss* pada *boss level*. Pada dasarnya *genre game* ini dapat dipecah lagi menjadi dua tipe yaitu *platformer puzzle* dan *action* (Bhosale, 2018). Dapat diambil contoh *game Limbo* pada Gambar 2.1 sebagai *game platformer puzzle* dan *Super Mario Bros* pada Gambar 2.2 sebagai *game platformer action*. *Super Mario Bros* adalah salah satu *game platformer* populer yang diterbitkan oleh *Nintendo*.



Gambar 2.1 *Limbo*

Sumber: (Sony Interactive Entertainment, 2018)

Gambar 2.2 *Super Mario Bros*

Sumber: (Tony Polanco, 2018)

Setelah rilisnya *Super Mario Bros* ke pasaran pada tahun 1985, banyak *game platformer* lain yang bermunculan khususnya *2D platformer* seperti *Sonic The Hedgehog* pada Gambar 2.3 di tahun 1994. *2D platformer game* termasuk *game side scrolling* yang yang berarti *game* tersebut dilihat dari samping sisi sudut kamera.

Gambar 2.3 *Sonic The Hedgehog*

Sumber: (Emuparadise, 2018)

Setelah popularnya *game* ber*genre 2D platformer* kemudian muncul *2D platformer* bertipe *endless runner* atau yang lebih dikenal dengan *2D endless runner*. *Game* ini merupakan *game* dimana pemain terus berjalan atau berlari tanpa henti melalui rintangan yang dihasilkan secara prosedural yang tak memilik akhir dan menuntut pemain untuk mendapatkan skor setinggi tingginya. *2D endless runner* banyak diminati karena kesederhanaan cara bermainnya terutama memudahkan pemain yang memainkannya pada perangkat *mobile*. Contoh *genre* ini adalah *Canabalt* (2009), *Temple Run* (2011), dan *JetPack Joyride* (2011) (Jos, 2014). Contoh dari *game Temple Run* ditampilkan pada Gambar 2.4. *Game* ini biasanya juga memiliki tingkat kesulitan yang semakin meningkat seiring bertambahnya skor dari pemain.

Gambar 2.4 *Temple Run*

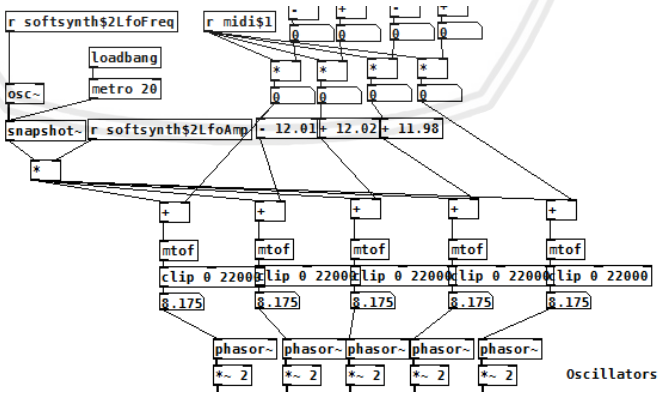
Sumber: (Waqas Ahmed, 2018)

2.2 *Dynamic audio*

Dynamic audio merupakan *audio* interaktif dalam game yang dapat langsung merespon masukan dari pemain dan membalasnya dengan *audio* yang dibutuhkan. Salah satu contoh penerapan *dynamic audio* yaitu saat melakukan gerakan meloncat pada karakter yang diam dan gerakan meloncat pada karakter yang berlari maka suara yang dikeluarkan akan beda atau suara langkah kaki pada saat karakter berlari dan berjalan, karena dalam praktiknya di dunia nyata tentu saja suara yang dihasilkan akan berbeda. *Dynamic audio* akan memberikan pengalaman bermain lebih natural dan imersif. Cara kerja *dynamic audio* sendiri adalah dengan mengolah suara langsung secara digital. Seperti contohnya menaikkan atau menurunkan frekuensi dari gelombang suara dari *asset* yang ada (Peerdeman, 2006).

2.3 *Pure Data*

Pure Data adalah bahasa *visual programming* untuk multimedia yang dikembangkan oleh Miller Puckette. Pemrograman dalam *Pure Data* menyerupai merancang dan membangun mesin untuk melakukan tugas yang diinginkan (Hancock, 2014). *Pure Data* memiliki cara *visual* untuk memprogram suara, yaitu dengan menghubungkan kotak - kotak kecil bersama dengan kabel *virtual*. Model pengkodean dalam *Pure Data* ditampilkan pada Gambar 2.5. *Pure Data* sejatinya dirancang untuk memungkinkan kolaborasi langsung di seluruh jaringan atau *internet*, memungkinkan musisi terhubung melalui LAN atau bahkan di bagian dunia yang berbeda untuk menciptakan musik bersama secara *real time*. *Pure Data* memiliki kelebihan yang mendukung *platform desktop*, *smartphone*, hingga *Raspberry Pi*.



Gambar 2.5 Pengkodean dalam *Pure Data*

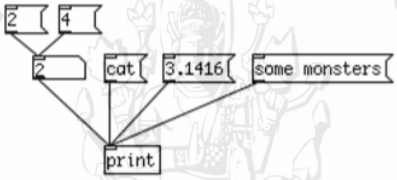
Sumber: (Andy Elmsley, 2018)

Tempat bekerja dalam *Pure Data* biasa disebut *patch* atau *canvas*. Di dalam *Pure Data* sendiri terdapat beberapa fungsi yaitu *object*, *message*, *number*,

symbol, dan *comment*. *Object* berfungsi untuk memberikan fungsi seperti *print* untuk mencetak *input*, *osc~* untuk memberikan fungsi osilator. Bentuk dari *object* digambarkan pada Gambar 2.6. Lalu ada *message* dan *number* untuk memberikan *input* pada perintah yang berbentuk tulisan dan angka. Bentuk dari *message* dan *numbe*r digambarkan pada Gambar 2.7. *Symbol* berfungsi untuk menyimpan informasi. *Comment* berfungsi untuk memberikan kutipan yang dapat digunakan sebagai panduan atau keterangan pada *canvas* (Holzer, 2012). Bentuk dari *symbol* dan *comment* digambarkan pada Gambar 2.8.

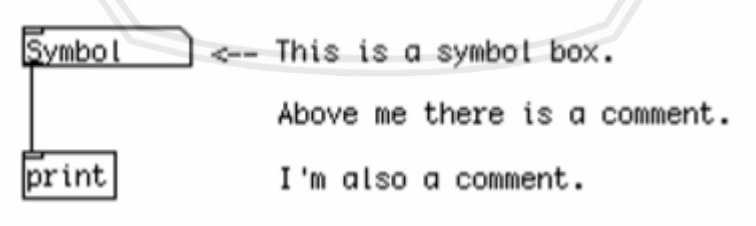
Gambar 2.6 Contoh *object*

Sumber: (Holzer, 2012)



Gambar 2.7 Contoh *message* dan *number*

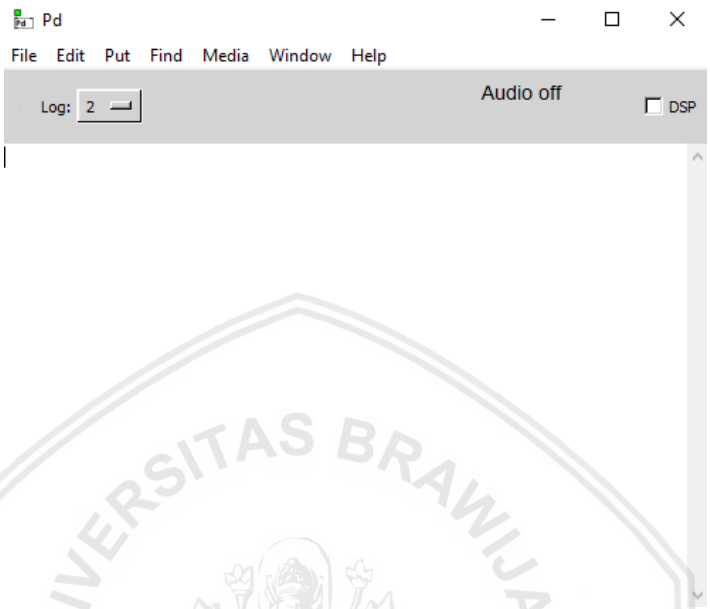
Sumber: (Holzer, 2012)



Gambar 2.8 Contoh *symbol* dan *comment*

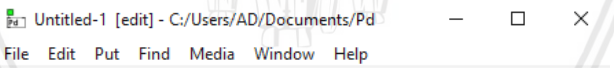
Sumber: (Holzer, 2012)

Berikut adalah contoh cara membuat *Pure Data patch* sederhana. Langkah pertama yang harus dilakukan adalah membuka aplikasi *Pure Data* hingga muncul tampilan seperti Gambar 2.9 berikut.



Gambar 2.9 Tampilan awal *Pure Data*

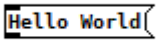
Kemudian membuat *canvas* dengan mengklik *File* lalu pilih menu *New*. Pada aplikasi akan muncul tampulan *canvas* seperti Gambar 2.10 berikut.



Gambar 2.10 Tampilan *canvas*

Selanjutnya adalah menempatkan *object*, *message*, dan lainnya dengan cara mengklik *Put* lalu pilih apa yang akan dimasukkan, di sini dicontohkan dengan memasukkan *message* bertuliskan "*Hello World*" seperti Gambar 2.11 berikut. Jika

tidak dapat melakukan apa apa, maka perlu masuk ke dalam *edit mode*. Masuk ke dalam *edit mode* dapat dilakukan dengan cara mengklik *edit* lalu centang *edit mode.*



Gambar 2.11 *Message* "*Hello Word*"

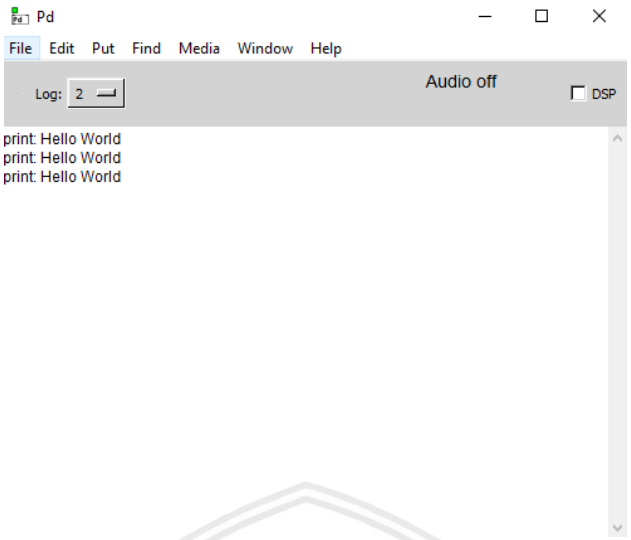
Selanjutnya adalah menambahkan *object print* dan *bang* untuk menjalankan perintah. Dan atur hingga seperti Gambar 2.12 berikut.

Gambar 2.12 Tampilan objek sebelum disambungkan

Langkah terakhir adalah menyambungkan titik hitam pada tiap objek agar tersambung satu sama lain seperti Gambar 2.13 berikut.

Gambar 2.13 Tampilan objek setelah disambungkan

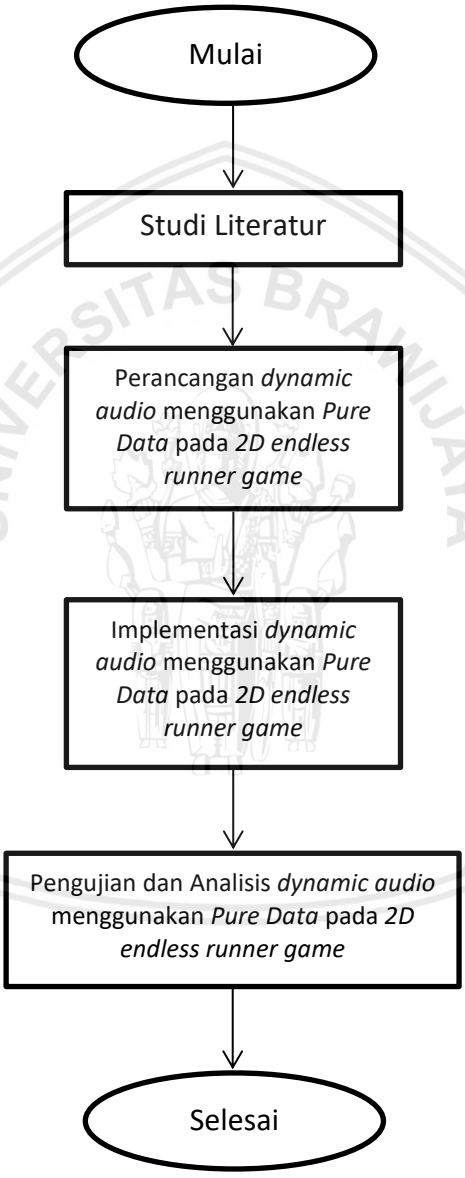
Setelah itu dapat langsung keluar dari *edit mode* dengan cara menghilangkan centang pada *edit mode* dan dapat langsung mengklik *bang* sebanyak yang diinginkan. Di sini akan diklik tombol *bang* sebanyak 3 kali, berarti akan muncul tulisan "*Hello World*" sebanyak 3 seperti pada Gambar 2.14 berikut.



Gambar 2.14 Tampilan keluaran

BAB 3 METODOLOGI

Gambar 3.1 merupakan diagram alir yang berisi tahapan yang akan dilakukan dalam metodologi penelitian. Penelitian ini bersifat implementatif dengan megimplementasikan *dynamic audio* menggunakan *Pure Data* pada *2D Endless Runner Game*. Metode penelitian yang digunakan adalah :



Gambar 3.1 Diagram alir metodologi

3.1 Studi Literatur

Studi literatur dilakukan untuk mempelajari dasar teori yang digunakan untuk penunjang penulisan. Pada tahap ini dilakukan pencarian terhadap berbagai sumber tertulis, baik berupa buku-buku, arsip, majalah, artikel, dan jurnal, atau dokumen-dokumen yang relevan dengan permasalahan yang dikaji. Sehingga informasi yang didapat dari studi kepustakaan ini dijadikan rujukan untuk memperkuat argumen yang ada. Studi literatur yang dipelajari adalah sebagai berikut :

- *2D Platformer game*.
- *Dynamic audio*.
- *Pure Data.*

3.2 Perancangan *dynamic audio* menggunakan *Pure Data* pada *2D endless runner game*

Tahap perancangan merupakan tahapan yang dilakukan untuk merancang pengimplementasian *dynamic audio* menggunakan *Pure Data* pada *2D endless runner game*. Tahap perancangan dimulai dari perancangan *dynamic audio* dan mempersiapkan *Unity3D*, *Pure Data*, dan *Heavy* untuk menyambungkan *Pure Data* dan *Unity3D*. Selanjutnya dapat langsung dilakukan koding *Pure Data* sesuai rancangan.

3.2.1 Perancangan *dynamic audio* pada *2D endless runner game*

Untuk perancangan *dynamic audio* pada *2D endless runner game*, akan ditentukan bagian mana yang akan diberi penerapan *dynamic audio*. Dapat diambil contoh yaitu bagian suara lari dan loncat dari karakter maupun pergantian lagu tiap suasana.

3.2.2 Perancangan *Pure Data*

Perancangan *Pure Data* dilakukan untuk membuat rancangan suara pada bagian *game* yang sudah ditentukan sebelumnya, bisa dalam bentuk efek suara maupun musik untuk latar belakang *game*. Perancangan *Pure Data* dilakukan dengan membuat rancangan langsung pada aplikasi *Pure Data*. Tahap ini juga dapat diartikan dengan membuat *synth module* atau *patch* pada *Pure Data*.

3.3 Implementasi *dynamic audio* menggunakan *Pure Data* pada *2D endless runner game*

Tahap implementasi mengacu dari hasil perancangan yang telah dilakukan. Dimulai dari perancangan *dynamic audio*, menyiapkan *Unity3D*, *Pure Data,* dan *Heavy* untuk menyambungkan *Pure Data* dan *Unity3D*, menentukan mekanisme mana yang akan diberi penerapan *dynamic audio*, dan melakukan koding sesuai rancangan.

Setelah *synth module* atau *patch* pada *Pure Data* selesai. Modul *Pure Data* tersebut perlu di*compile* menggunakan *Heavy* agar menjadi kode *C# native* yang dapat langsung diimplementasikan pada *Unity3D*. Setelah selesai dapat langsung dilakukan koding pada *Unity3D.*

3.4 Pengujian dan Analisis *dynamic audio* menggunakan *Pure Data* pada *2D endless runner game*

Tahap pengujian bertujuan untuk mengetahui apakah hasil implementasi *dynamic audio* menggunakan *Pure Data* pada *2D endless runner game* sudah sesuai dengan perancangan yang telah dijabarkan pada tahap sebelumnya. Pengujian dilakukan dengan melihat aksi reaksi dari audio dan mekanismenya apakah sudah sesuai dengan perancangan yang telah ditentukan sebelumnya. Lalu membandingkan performa antara *dynamic audio* dan *static audio* pada bagian FPS yang dihasilkan. Pengujian FPS ditujukan untuk mengetahui apakah penerapan *dynamic audio* memberikan dampak performa pada 2D *endless runner game*.

BAB 4 PERANCANGAN

4.1 Perancangan *dynamic audio* pada *2D endless runner game*

Pada tahap ini akan ditentukan bagian mana yang akan diberi penerapan *dynamic audio*. Di sini ditentukan bagian-bagian pada *2D endless runner game* yang akan diberi penerapan *dynamic audio*. Bagian tersebut adalah :

1. ***Background music main menu*.**

   Pada bagian ini akan dibuat musik latar pada tampilan *main menu.*

2. **Efek suara *player*.**

   Bagian ini dibagi menjadi 3 bagian yaitu :

   - Efek suara saat karakter lari, suara lari awal akan berubah menjadi suara yang memiliki frekuensi lebih tinggi saat karakter melakukan gerakan maju.
   - Efek suara saat karakter loncat, efek suara ini akan mengubah frekuensi sesuai tinggi loncatan karakter.
   - Efek suara *hit*, yaitu adalah efek suara saat nyawa karakter berkurang, hal ini terjadi karena karakter menabrak rintangan, jatuh, atau tertabrak oleh musuh.

3. **Efek suara lingkungan.**

   Bagian ini dibagi menjadi 2 bagian yaitu :

   - Efek suara angin, efek suara ini akan digunakan pada saat latar belakang *game* siang.
   - Efek suara jangkrik, efek suara ini akan digunakan pada saat latar belakang *game* malam.

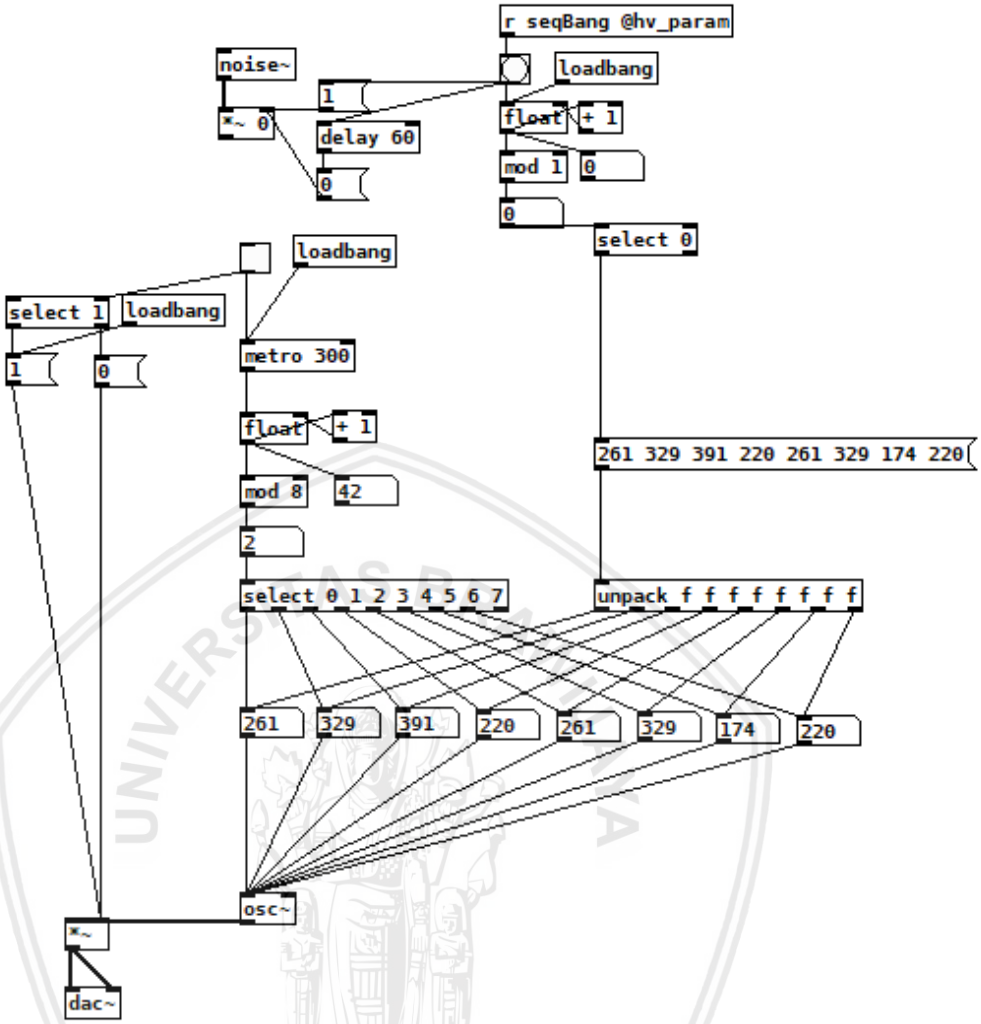
4.2 Perancangan *Pure Data*

Pada tahap ini akan dijelaskan bagaimana merancancang semua *asset* suara yang berupa *Pure Data patch* sesuai bagian di dalam *game* yang telah ditentukan sebelumnya. Perancangan *Pure Data patch* dilakukan langsung di dalam aplikasi *Pure Data*.

4.2.1 *Patch background music main menu,* efek suara lari, dan efek *hit*.

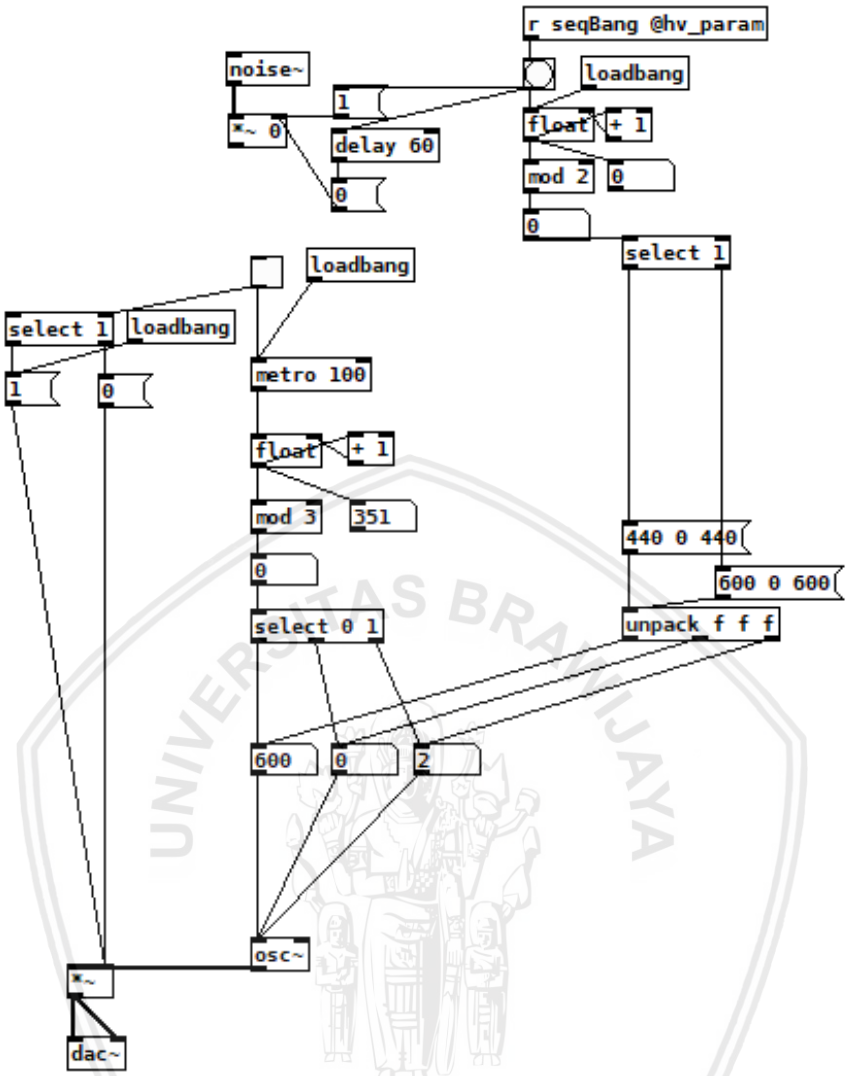
Pada bagian ini akan dilakukan pembuatan *Pure Data patch* yang akan digunakan untuk *background music main menu*, efek suara lari, dan efek *hit* pada *game 2D endless runner game.* Ketiga *patch* tersebut memiliki kesamaan model sehingga dapat dimasukkan 3 *patch* tersebut pada satu pembahasan.

Musik bagian *main menu* akan langsung dimainkan ketika *game* dibuka dan telah berhasil memasuki *main menu.* Perancangan *background music main menu* ditampilkan pada Gambar 4.1.



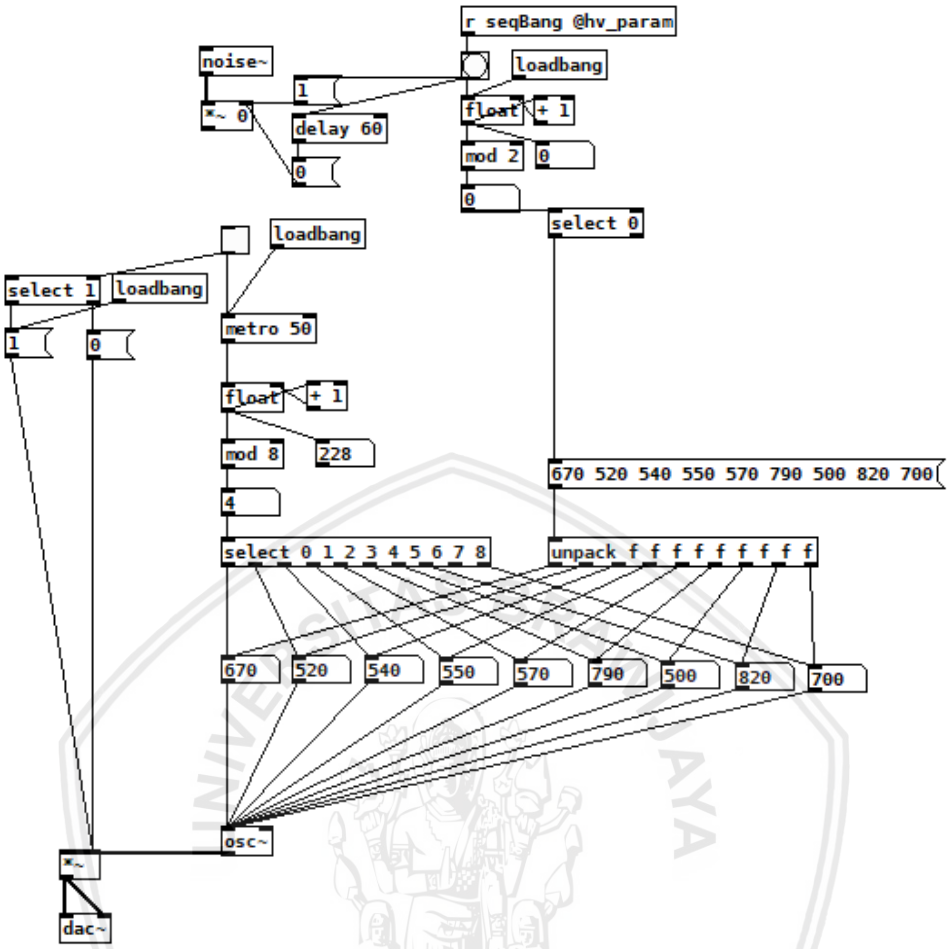
Gambar 4.1 Perancangan *background music main menu*

Kemudian dilanjutkan dengan *Pure Data patch* efek suara lari dan efek *hit.* Kedua efek ini akan dimainkan ketika memasuki *state* tertentu. Efek suara lari memiliki dua tatanan frekuensi yang dapat dimainkan. Tatanan frekuensi pertama atau frekuensi rendah akan langsung dimainkan ketika memasuki game untuk suara karakter berlari biasa, sedangkan tatanan frekuensi kedua atau frekuensi tinggi akan dimainkan ketika karakter melakukan gerakan maju. Perancangan efek suara lari ditampilkan pada Gambar 4.2.



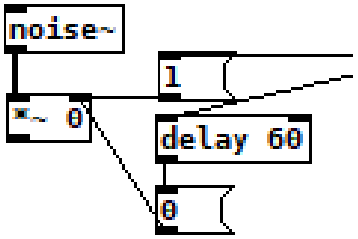
Gambar 4.2 Perancangan efek suara lari

Efek suara *hit* akan dimainkan ketika nyawa dari karakter berkurang karena menabrak rintangan, jatuh, atau tertabrak oleh musuh. Efek suara ini akan dimainkan dengan sangat cepat agar suara tidak dimainkan terlalu lama. Perancangan efek suara *hit* ditampilkan pada Gambar 4.3.



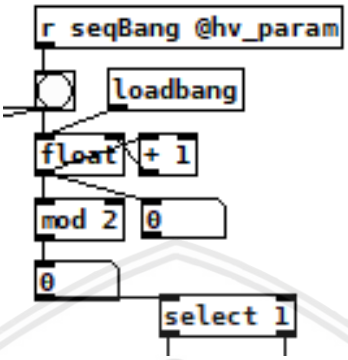
Gambar 4.3 Perancangan efek suara *hit*

Ketiga *patch* tersebut memiliki kesamaan model, yaitu kesamaan pada fungsinya yang memainkan frekuensi berurutan yang telah ditulis pada masing – masing *patch. Patch* tersebut terdiri dari beberapa fungsi. Yang pertama adalah fungsi *noise* pada Gambar 4.4. Bagian ini berfungsi untuk memberikan efek *noise* dan *delay* pada *Pure Data patch* yang telah dirancang.



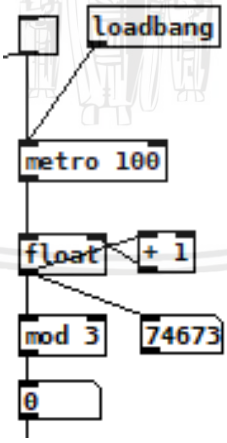
Gambar 4.4 Fungsi *noise*

Selanjutnya adalah fungsi *bang.* Bagian ini berfungsi untuk menentukan frekuensi mana yang akan dimainkan. Pada bagian ini juga terdapat fungsi *seqBang @hv_param* yang bertujuan agar fungsi dapat dikenali pada *Unity3D.* Bagian ini ditampilkan pada Gambar 4.5.



Gambar 4.5 Fungsi *bang*

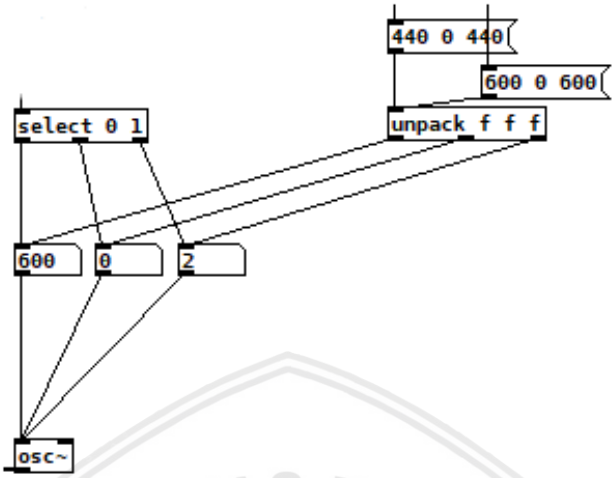
Kemudian dilanjutkan fungsi metronom. Metronom adalah sebuat alat yang dapat mengeluarkan suara dengan ketukan stabil. Pada patch ini metronom berfungsi untuk memberikan ketukan stabil yang kemudian akan dibaca oleh bang yang akan menghasilkan bang terus menerus untuk membunyikan frekuensi per frekuensi yang telah disiapkan. Fungsi metronom ditampilkan pada Gambar 4.6.



Gambar 4.6 Fungsi metronom

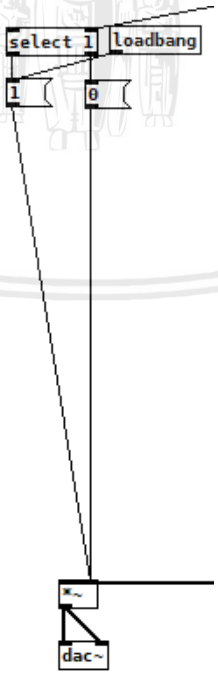
Tahap selanjutnya adalah bagian utama dari musik, yaitu bagian dimana frekuensi yang dimainkan dari musik atau efek terbentuk. Bagian ini adalah tempat dimana frekuensi musik atau efek disimpan. Bagian ini ditampilkan pada Gambar 4.7. Bagian kanan dari fungsi frekuensi adalah bagian dimana pilihan frekuensi disimpan, sedangkan bagian kiri adalah frekuensi yang siap dimainkan. Pada

bagian ini juga terdapat fungsi *osc~* atau osilator yang berfungsi untuk membunyikan frekuensi yang telah ditentukan.



Gambar 4.7 Fungsi frekuensi

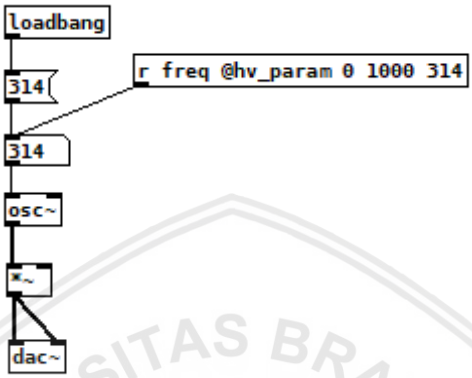
Yang terakhir adalah fungsi *select.* Fungsi ini ditambahkan agar suara keluaran dari *patch* dapat dimatikan atau dinyalakan. Dalam fungsi ini juga terdapat *dac~* yang akan membuat *patch* dapat dibunyikan melalui *speaker*. Bagian ini ditampilkan pada Gambar 4.8.



Gambar 4.8 Fungsi *select*

4.2.2 *Patch* efek suara loncat

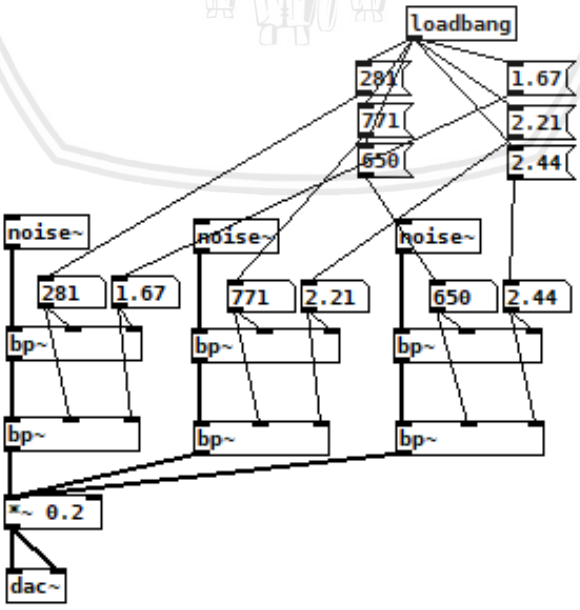
Pada bagian ini akan dilakukan pembuatan *Pure Data patch* yang akan digunakan untuk efek suara loncat pada *game 2D endless runner game. Patch* efek suara ini memiliki fungsi *osc~* dan *dac~* sebagaimana telah dijelaskan sebelumnya. Bagian ini memiliki *freq @hv_param* yang digunakan untuk pembacaan frekuensi pada unity. *Patch* efek suara loncat ditampilkan pada Gambar 4.9.



Gambar 4.9 Perancangan efek suara loncat

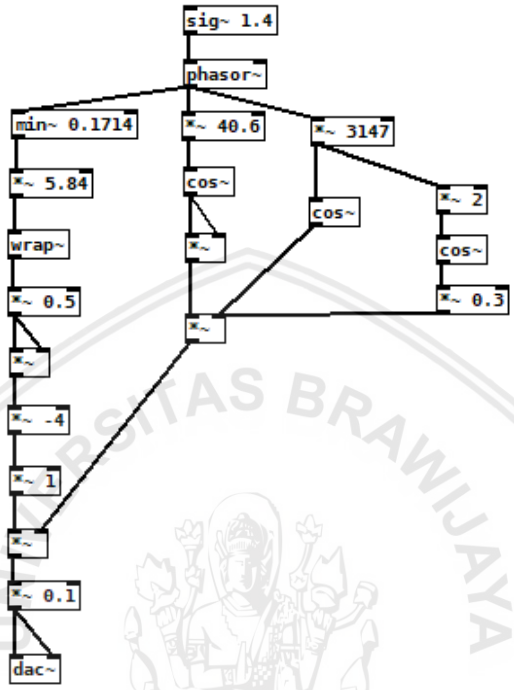
4.2.3 *Patch* efek suara lingkungan

Pada bagian ini akan dilakukan pembuatan *Pure Data patch* yang akan digunakan untuk efek suara angin pada *game 2D endless runner game.* Efek suara ini dibuat menggunakan fungsi frekuensi yang disambungkan kepada fungsi *noise*. *Patch* efek suara angin ditampilkan pada Gambar 4.10.



Gambar 4.10 Perancangan efek suara angin

Bagian terakhir adalah pembuatan *Pure Data patch* efek suara jangkrik. Efek suara ini menggunakan fungsi *sig~* yang berfungsi untuk mengubah angka menjadi sinyal suara dan *phasor~* untuk menghasilkan gelombang suara yang pada akhirnya akan langsung disambungkan pada *dac~* untuk mengeluarkan suara pada *speaker. Patch* efek suara jangkrik ditampilkan pada Gambar 4.11.

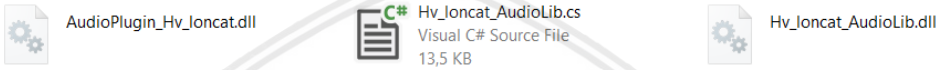


Gambar 4.11 Perancangan efek suara jangkrik

BAB 5 IMPLEMENTASI

5.1 Implementasi *Heavy*

Bagian ini akan membahas tentang pengimplementasian *Pure Data patch* yang telah dirancang sebelumnya kepada *Heavy*. Tahap ini dilakukan dengan cara melakukan *compile* terhadap *Pure Data patch* kepada *compiler Heavy*. Di sini dicontohkan hasil *compile* dari *Pure Data patch* loncat yang telah dirancang pada tahap sebelumnya. Hasil dari *compile Pure Data patch* menggunakan *Heavy* ditampilkan pada Gambar 5.1 Hasil *compile Pure Data patch* efek suara loncat Gambar 5.1.



Gambar 5.1 Hasil *compile Pure Data patch* efek suara loncat

Dari hasil *compile Pure Data patch* menggunakan *Heavy* dapat dilihat bahwa ada 3 *file* yang dihasilkan, yaitu *AudioPlugin_Hv_loncat.dll, Hv_loncat_AudioLib.cs*, dan *Hv_loncat_AudioLib.dll*. Ketiga *file* tersebut merupakan *plugin* dari *Unity3D* dan siap untuk langsung dimasukkan ke dalam *Unity3D*.

5.2 Implementasi *Unity3D*

Bagian terakhir dari implementasi adalah melakukan koding pada *Unity3D*. Bagian ini dilakukan dengan melakukan koding untuk menentukan pada saat apa musik atau efek suara akan dibunyikan. Di sini dicontohkan implementasi efek suara loncat dengan membuat *script* pada *Unity3D*. *Script* bagian efek suara loncat dipilih karena sudah mewakili *script* suara lainnya. *Script* tersebut bernama *ControlScript.cs* yang berfungsi untuk menentukan bagaimana efek suara loncat akan dimainkan sesuai dengan rancangan sebelumnya. Kode *script* efek suara loncat ditampilkan pada Tabel 5.1.

Tabel 5.1 Kode *script* efek suara loncat

ControlScript.cs

```
1   using UnityEngine;
2   using System.Collections;
3
4   public class ControlScript : MonoBehaviour
5   {
6       public Hv_loncat_AudioLib HeavyScript;
7       public AudioSource audioSource;
8
9       [Range(1, 20)]
10      public float freqChangeRate;
11
```

```
12      [Range(1, 1000)]
13      public float standardFreq;
14
15      [Range(1, 1000)]
16      public float maxFreq;
17
18      bool falling, grounded = true;
19
20      void Start()
21      {
22
23      }
24
25      void Update()
26      {
27          if (Input.GetKey(KeyCode.Space))
28          {
29              if (Input.GetKeyDown(KeyCode.Space) && grounded)
                {
30                  audioSource.Play();
31                  HeavyScript.freq = standardFreq;
32                  grounded = false;
33              }
34              if (HeavyScript.freq < maxFreq)
35              {
36                  HeavyScript.freq += freqChangeRate / 2;
37              }
38          }
39
40          if (Input.GetKeyUp(KeyCode.Space))
41          {
42              falling = true;
43          }
44          if (falling)
45          {
46              HeavyScript.freq -= freqChangeRate;
47              if (HeavyScript.freq <= standardFreq)
48              {
49                  grounded = true;
50                  falling = false;
51              }
52          }
53
54          if (grounded)
55          {
56              audioSource.Stop();
57          }
58
59          if (Input.GetKeyDown(KeyCode.A))
60          {
61              grounded = true;
62          }
63      }
64  }
65
```

Kode *script* efek suara loncat memanfaatkan logika *if-else* untuk memainkan atau mematikan efek suara loncat. Di sini diberikan fungsi pada baris 9 – 16 untuk mengatur minimal dan maksimal frekuensi, *rate* dari perubahan frekuensi, serta frekuensi standar yang dapat langsung diubah menggunakan *slider* pada *Unity3D*. Tujuan menambahkan fitur tersebut agar tinggi rendah frekuensi dari efek suara loncat dapat diubah dan lebih dinamis.

Di sini juga terdapat fungsi *grounded* dan *falling* pada baris 18. *Grounded* adalah fungsi saat karakter berada pada *state* lari atau dapat disebut ada pada tanah maka efek suara loncat tidak akan dimainkan. Fungsi tersendiri dari *grounded* ada pada baris 55 – 63, sedangkan efek *falling* merupakan efek dimana frekuensi suara akan kembali pada frekuensi standar saat karakter selesai mencapai titik loncatan tertentu, juga dapat dikatakan saat karakter kembali ke tanah setelah melakukan loncat. Fungsi tersendiri dari *falling* ada pada baris 45 – 53.

BAB 6 PENGUJIAN

6.1 Pengujian Validasi

Pengujian validasi dilakukan dengan tujuan untuk menguji apakah bagian dari *2D endless runner game* yang telah diberi *dynamic audio* dapat berjalan sesuai rancangan yang telah ditentukan sebelumnya.

Tabel 6.1 Pengujian validasi *background music main menu* 1

Pengujian Validasi *Background Music Main Menu*	
Objective	*Background music main menu* dapat langsung dimainkan ketika *game* dibuka.
Status	Valid

Tabel 6.2 Pengujian validasi *background music main menu* 2

Pengujian Validasi *Background Music Main Menu*	
Objective	*Background music main menu* berhenti ketika masuk ke dalam *in-game*.
Status	Valid

Tabel 6.3 Pengujian validasi efek suara lari 1

Pengujian Validasi Efek Suara Lari	
Objective	Efek suara lari dengan frekuensi rendah akan langsung dimainkan ketika *in-game*.
Status	Valid

Tabel 6.4 Pengujian validasi efek suara lari 2

Pengujian Validasi Efek Suara Lari	
Objective	Efek suara lari tinggi akan dimainkan ketika karakter melakukan gerakan maju.
Status	Valid

Tabel 6.5 Pengujian validasi efek suara lari 3

Pengujian Validasi Efek Suara Lari	
Objective	Efek suara lari akan berhenti ketika karakter melakukan loncat.
Status	Valid

Tabel 6.6 Pengujian validasi efek suara loncat 1

Pengujian Validasi Efek Suara Loncat	
Objective	Efek suara loncat akan dimainkan ketika karakter meloncat. Semakin tinggi karakter meloncat maka akan semakin tinggi frekuensi yang dibunyikan.
Status	Valid

Tabel 6.7 Pengujian validasi efek suara loncat 2

Pengujian Validasi Efek Suara Loncat	
Objective	Frekuensi efek suara loncat akan kembali pada frekuensi awal ketika karakter turun sesudah melakukan loncat.
Status	Valid

Tabel 6.8 Pengujian validasi efek suara *hit*

Pengujian Validasi Efek Suara *Hit*	
Objective	Efek suara *hit* akan dimainkan ketika nyawa karakter berkurang dengan menabrak rintangan, jatuh, atau tertabrak oleh musuh.
Status	Valid

Tabel 6.9 Pengujian validasi efek suara angin

Pengujian Validasi Efek Suara Angin	
Objective	Efek suara angin akan dimainkan langsung ketika kondisi pada *in-game* adalah siang.
Status	Valid

Tabel 6.10 Pengujian validasi efek suara jangkrik

Pengujian Validasi Efek Suara Jangkrik	
Objective	Efek suara jangkrik akan dimainkan langsung ketika kondisi pada *in-game* adalah malam.
Status	Valid

Dari pengujian validasi yang telah dilakukan, didapatkan hasil valid pada semua skenario pengujian. Maka dari itu dapat dikatakan bahwa implementasi *dynamic audio* menggunakan *Pure Data* pada *2D endless runner game* sudah sesuai pada rancangan sebelumnya.

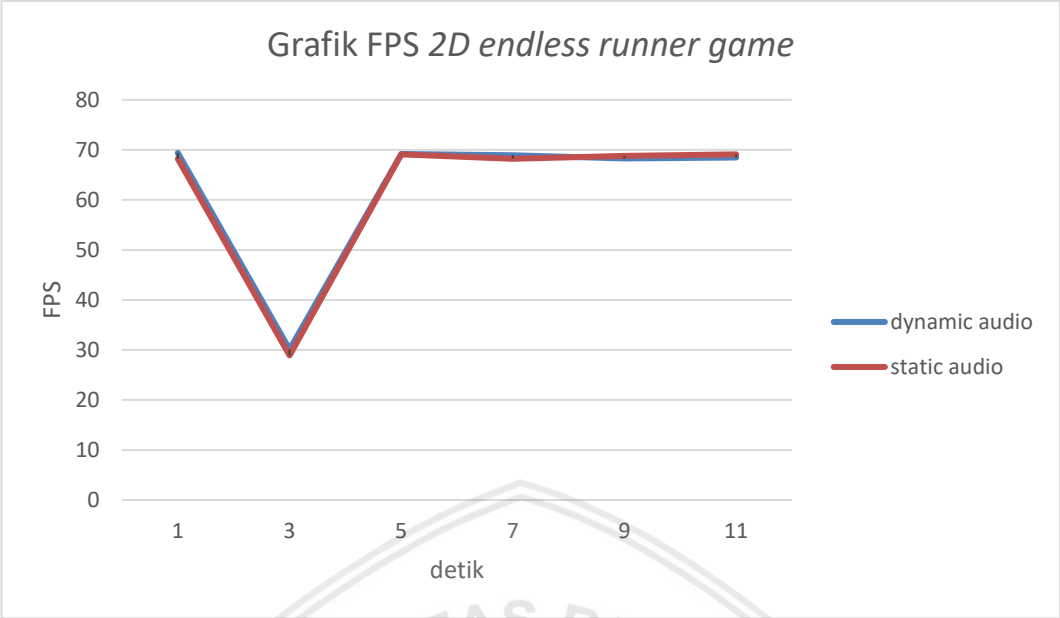
6.2 Pengujian FPS

Pengujian FPS dilakukan dengan tujuan untuk menguji apakah terjadi perbedaan performa dari *2D endless runner game* yang telah diberi *dynamic audio* dengan *2D endless runner game* yang menggunakan *asset* MP3 atau *static audio*. Pengujian FPS dilakukan dengan cara melihat pada fungsi *stats* yang ada pada *Unity3D*. Pengujian ini dilakukan dengan skenario yang sama pada detik yang telah ditentukan. Skenario tersebut dapat dilihat pada Tabel 6.11.

Tabel 6.11 Skenario pengujian

Detik	Skenario pengujian
1	*Background music main menu*
3	Masuk *gameplay* (efek suara lari dan angin otomatis diputar)
5	Efek suara loncat
7	Efek suara lari cepat
9	Efek suara hit
11	Efek suara jangkrik

Skenario pengujian di atas dilakukan dengan menjalankan *2D endless runner game* yang telah diberi *dynamic audio* dan *2D endless runner game* yang menggunakan *asset* MP3 atau *static audio* secara bersamaan. Hasil pengujian performa dari *2D endless runner game* ditampilkan pada Gambar 6.1 dan Tabel 6.12. Pada skenario pengujian tersebut, *dynamic audio* menggunakan 6 *asset* sedangkan pada *static audio* memiliki 7 *asset*, hal itu disebabkan oleh suara efek lari biasa dan efek suara lari cepat dapat dipenuhi oleh 1 *asset dynamic audio* saja.



Gambar 6.1 Grafik FPS *2D endless runner game*

Tabel 6.12 Pengujian FPS

Asset	***Min.* FPS**	***Avg.* FPS**	***Max.* FPS**
Dynamic Audio	30,1	62,4	69,4
Static Audio	28,9	62,1	69,1

Dari pengujian FPS yang telah dilakukan dapat dilihat bahwa performa dari *2D endless runner game* yang telah diberi *dynamic audio* dengan *2D endless runner game* yang menggunakan *asset* MP3 atau *static audio* memiliki perbedaan yang tidak signifikan. Pada detik ketiga terjadi penurunan FPS yang signifikan baik dari *2D endless runner game* yang telah diberi *dynamic audio* maupun *2D endless runner game* yang menggunakan *asset* MP3 atau *static audio.* Penurunan FPS tersebut dikarenakan adanya pergantian dari *scene main menu* kepada *scene gameplay*. Dapat disimpulkan bahwa penerapan *dynamic audio* menggunakan *Pure Data* pada *2D endless runner game* tidak berpengaruh pada performa *game* dan memberikan kelebihan terhadap audio yang lebih dinamis. Dinamis yang dimaksud yaitu pada *asset dynamic audio* dapat diubah frekuensi, tempo, dan komponen suara lainnya agar dapat mengubah suara tanpa harus membuat *asset* baru untuk *static audio*.

BAB 7 KESIMPULAN

7.1 Kesimpulan

Penelitian tentang penerapan *dynamic audio* menggunakan *Pure Data* pada *2D endless runner game* telah berhasil dilakukan sesuai dengan metode yang telah ditentukan sebelumnya. Hasil dari penelitian ini dapat disimpulkan sebagai berikut :

1. Memanfaatkan *Pure Data* untuk penerapan dynamic audio *pada 2D endless runner game* memberikan kebebasan pengembang *game* untuk mengubah suara efek maupun musik tanpa harus mencari atau membuat *asset* suara baru lagi.
2. Dari pengujian FPS yang telah dilakukan dapat dilihat bahwa kinerja dari *dynamic audio* menggunakan *Pure Data* pada *2D endless runner game* tidak memiliki perbedaan yang signifikan terhadap kinerja *game* yang menggunakan *asset static audio* atau MP3. Pada pengujian yang telah dilakukan didapatkan rata-rata FPS untuk *2D endless runner game* yang telah diberi *dynamic audio* mendapatkan hasil 62,4 FPS sedangkan untuk *2D endless runner game* yang menggunakan *asset* MP3 atau *static audio* mendapatkan rata-rata 62,1 FPS. Dengan demikian dapat disimpulkan bahwa penerapan *dynamic audio* menggunakan *Pure Data* pada *2D endless runner game* tidak berpengaruh pada performa *game*.

7.2 Saran

Berdasarkan pada permasalahan yang diangkat oleh penulis yaitu mengenai penerapan *dynamic audio* menggunakan *Pure Data* pada *2D endless runner game*, maka dari itu penulis memberikan saran sebagai berikut :

1. Penulis menerapkan *dynamic audio* menggunakan *Pure Data* pada *2D endless runner game* yang memiliki *platform Windows*. Mengingat *Pure Data* memiliki kompatibilitas yang hampir mendukung semua *platform*, dapat dilakukan penelitian dengan menerapkan *Pure Data* pada *platform* lainnya.
2. Pada penelitian yang telah dilakukan, penulis menggunakan metode *Heavy* untuk mengimplementasikan *Pure Data* pada *Unity3D*. *Heavy* bukan satu – satunya metode untuk mengimplementasikan *Pure Data* pada *Unity3D*. Dapat dilakukan pengimplementasian dengan metode *libpd* untuk mengimplementasikan *Pure Data* pada *Unity3D*.

DAFTAR REFERENSI

Andy Elmsley, 2018. *How to Use Pure Data in Unity*. [online] Tersedia di: <http://melodrive.com/blog/how-to-use-pure-data-in-unity/> [Diakses 31 Agustus 2018].

Bhosale, T., 2018. 2D Platformer Game In Unity Engine. *International Research Journal of Engineering and Technology*, [e-journal] 5(4) pp. 3021-3024. Tersedia di: < https://www.irjet.net/archives/V5/i4/IRJET-V5I4667.pdf> [Diakses 28 Agustus 2018].

Emuparadise, 2018. *Sonic the Hedgehog 3 (USA) ROM*. [online] Tersedia di: <https://www.emuparadise.me/Sega_Genesis_-_Sega_Megadrive_ROMs/Sonic_the_Hedgehog_3_(USA)/39140> [Diakses 30 Agustus 2018].

Gustafsson, A., 2014. *Bachelor Thesis in Computer Science An Analysis of Platform Game Design*. [e-book] Department of Computer Science. Tersedia melalui: <https://www.diva-portal.org/smash/get/diva2:728079/FULLTEXT01.pdf> [Diakses 28 Agustus 2018].

Hancock, O., 2014. Learning of Pure Data : A Case Study. *Journal of Music, Technology & Education*, [e-journal] 7(1), pp. 93–112. Tersedia di: <https://www.researchgate.net/publication/262583696_Play-based_constructionist_learning_of_Pure_Data_A_case_study> [Diakses 27 Agustus 2018].

Holzer, D., 2012. *Pure Data*. [e-book] Floss Manuals. Tersedia melalui: <https://www.faa.gov/data_research/aviation/aerospace_forecasts/media/FY2017-37_FAA_Aerospace_Forecast.pdf%0Ahttp://xlink.rsc.org/?DOI=C4OB02655>. [Diakses 2 Oktober 2018].

Iftikhar, 2015. *An Automated Model Based Testing Approach For Platform Games*. Quest Lab, National University of Computer and Emerging Sciences. Islamabad, Pakistan, 30 September - 2 Oktober 2015. 2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems, MODELS 2015.

Jos, 2014. *Procedural Level Balancing in Runner Games*. Porto Alegre, Brazil, 12 - 14 November 2014. SBC - Proceedings of the SBGames 2014.

Peerdeman, P., 2006. *Sound and Music in Games*. [pdf] VU Amsterdam. Tersedia di: <http://www.peterpeerdeman.nl/vu/ls/peerdeman_sound_and_music_in_games.pdf> [Diakses 16 September 2018].

Sony Interactive Entertainment, 2018. *LIMBO on PS4*. [online] Tersedia di: <https://store.playstation.com/en-us/product/UP2054-CUSA01664_00-PLAYDEADD11LIMBO> [Diakses 30 Agustus 2018].

Tony Polanco, 2018. *Mint-Condition Copy of Super Mario Bros. Sold for $30,000 - Geek.com*. [online] Tersedia di: <https://www.geek.com/games/mint-condition-copy-of-super-mario-bros-sold-for-30000-1709707/> [Diakses 30 Agustus 2018].

Waqas Ahmed, 2018. *Temple Run [Review]*. [online] Tersedia di: <https://www.addictivetips.com/mobile/temple-run-for-android-iphone-ipad-ipod-touch-review/> [Diakses 30 Agustus 2018].