Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 3, No. 1, Januari 2019, hlm. 1093-1099 http://j-ptiik.ub.ac.id

Implementasi *Pure Data Programming* untuk *Generate Real-Time Game Audio Synthesis* pada *Platformer Game*

Bariq Najmi Rizqullah Kartiko Putro[1], Eriq Muhammad Adams Jonemaro[2], Tri Afirianto[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]bariq3396@gmail.com, [2]eriq.adams@ub.ac.id, [3]tri.afirianto@ub.ac.id

Abstrak

Audio memainkan peran yang sangat penting di dalam suatu permainan selain dari komponen lain pengembangan sebuah permainan. Dengan menggunakan proses perekaman, hasil yang didapatkan adalah serangkaian gelombang amplitudo, yang mana mengandung suara-suara dari hasil perekaman. Hasil dari perekaman tersebut menghasilkan suatu data. Ruang penyimpanan yang dibutuhkan untuk menyimpan sebuah data suara yaitu, setiap 1 menit dari suara *stereo* dengan *sample rate* 44,1 kHz membutuhkan 10 *megabyte* ruang penyimpanan. Terdapat teknologi *audio* yang dapat dimanfaatkan dalam pengembangan permainan. *Pure Data* merupakan bahasa pemrograman visual untuk tujuan multimedia. *Pure Data* dapat digunakan untuk proses dan menghasilkan suara, *video*, grafis dua Dimensi atau tiga dimensi, dan sebuah MIDI (Musical Intrument Digital Interface) dalam kepentingan insinyur *audio*. Selain itu, terdapat kerangka kerja alat bantu *Heavy* untuk menghasilkan *audio* tambahan menjembatani implementasi dari *Pure Data* ke *Unity*. *Heavy* menggunakan prinsip perangkat lunak modern untuk menghasilkan kode. Pada penelitian kali ini dibahas mengenai efek suara dan musik latar. Semua aset suara yang dihasilkan akan diimplementasikan ke dalam permainan. Hasil dari penelitian ini adalah perbandingan ukuran dokumen dari ketiga jenis suara yaitu, efek suara, musik latar, dan musik menu utama. *Pure Data* menghasilkan prosedural *audio* yang merupakan jenis *audio* dinamis. Kelebihan dari *audio* dinamis adalah efek suara yang dihasilkan memberikan kontribusi keterlibatan emosional kepada pemain. Kerangka kerja *Heavy* digunakan untuk mengimplementasikan *Pure Data* ke *Unity*. Terdapat dua pengujian untuk menguji *Pure Data* yaitu, dengan pengujian unit dan validasi.

Kata kunci: *Pure Data, Unity, Heavy Audio Tools,* Efek Suara, Musik Latar

Abstract

Audio plays a very important role in the game besides the another component of game development. By using the recording processs, the result obtained are a series of amplitude waves, which contain sounds from recording. The result of the recording produce a data. The storage space needed to store a sound data is every 1 minute of stereo sound with a 44,1 kHz sample rate requires 10 megabyte of storage space. There is an audio technology that can be used in game development. Pure Data is visual programming language for multimedia purpose. Pure Data can be used to process and generate sound, video, 2D or 3D graphics and MIDI for sound engineering purpose. Furthermore, there is framework called Heavy for easily generating audio plugins, used in implementating Pure data to Unity. Heavy make use of modern software pronciples to generate highly optimized code. In this study discussed the sound effects and background music. Sound assets will be implemented into the game. The result of this study are the comparison of document sizes of the three types of sound, sound eefect, background music, and main menu music. Pure Data generate procedural audio which is a dynamic type of audio. The advantage of dynamic audio is sound effects that contribute to emotional involvement of the player. Heavy framework is used to implement Pure Data to unity. There are two types of testing Pure Data, unit testing and validation.

Keywords: *Pure Data, Unity, Heavy Audio Tools, Sound Effect, Background Music*

1. PENDAHULUAN

Audio yang terdapat di dalam suatu *video game* merupakan fitur interaktif yang dipresentasikan, yaitu sebagai aksi yang memicu keluarnya sebuah dialog, efek suara, *ambient music*, hingga musik latar. Secara tradisional teknologi audio dihasilkan dengan proses perekaman atau *recording*. Sinyal suara di dunia nyata ditangkap melalui mikrofon, selanjutnya terdapat proses *mixing*, dan *mastering* untuk dirubah kedalam bentuk akhir dari *audio* tersebut (Farnell, 2007). Setiap bagian musik dan efek suara dipastikan diciptakan melalui langkah ini. Dengan proses perekaman, hasil yang didapatkan adalah serangkaian gelombang amplitudo. Gelombang tersebut mengandung suara-suara dari hasil perekaman dan hasil perekaman tersebut menghasilkan suatu data. Contohnya adalah data *audio* dengan format *mp3*. Banyak ruang penyimpanan yang dibutuhkan untuk menyimpan sebuah data suara yaitu, setiap 1 menit dari suara *stereo* dengan *sample rate* 44,1 kHz membutuhkan 10 *megabyte* ruang penyimpanan (Gamasutra, 1997).

Terdapat alternatif yang dapat memudahkan pembuatan *audio*, suara orisinal dan kreasi musik, di dalam *video game* dalam bentuk teknologi lingkungan pengembangan terintegrasi. *Pure Data* merupakan salah satu teknologi *audio* yang handal serta cakupan penggunanya cukup luas. *Pure Data* adalah bahasa pemrograman visual yang *open source* ditujukan untuk multimedia menjadi sebuah lingkungan pengembangan sehingga dapat digunakan sebagai *sound engine* di dalam aplikasi *mobile*, permainan, halaman sebuah *web*, dan proyek seni. Penggunaan dari *Pure Data* sangatlah mudah dan sederhana, selain itu dapat berjalan fleksibel sesuai kapabilitas *Pure data* itu sendiri. Contoh seberapa fleksibel *Pure Data* adalah penerapan fungsinya di dalam *mobile apps* seperti *iOS* dan *Android*, hingga musik interaktif di dalam pengembangan *game* (Kirn, 2012).

Pure Data memberi kemudahan dalam menghasilkan *audio*. Pure *Data* mempunyai kelebihan dibandingkan dengan teknologi *audio* tradisional lainnya. *Pure Data* menghasilkan prosedural *audio* yang merupakan jenis audio yang dinamis (*dynamic*). *Audio* dinamis dapat didefinisikan menjadi dua bentuk yang berbeda yaitu interaktif (*interactive*) *audio* dan adaptif (*adaptive*) *audio*.

Penelitian ini membahas tentang implementasi bahasa pemrograman visual *Pure Data* yang dimana pada penelitian ini difokuskan untuk pembuatan prosedural *audio* di dalam studi kasus *video game* yang ber*genre Platformer* yang mana mekanik dalam permainan ini pemain membimbing atau mengontrol karakter untuk melompati panggung (*platform*), hambatan, dan rintangan (Greenslade, 2006). Jenis suara yang difokuskan pada penelitian ini adalah *synthesis*. *Synthesis* sendiri merupakan proses dari penggabungan kumpulan acak suara elektronik yang nantinya memiliki keluaran suara yang beragam sesuai dengan kreativitas pembuatnya (Russ, 2009).

Hasil dari penelitian ini berupa *demo* permainan yang memanfaatkan *Pure data* dan kerangka kerja (*framework*) *Heavy* dalam pembuatan prosedural *audio* yang terkandung di dalamnya. Diharapkan dengan adanya penelitian ini penulis ikut menambah kajian yang lebih difokuskan ke arah *audio* permainan di dalam pengembangan sebuah *video game*.

2. DASAR TEORI

2.1. *Pure Data*

Pure Data (atau hanya *Pd*) adalah sebuah lingkungan bahasa pemrograman *visual* untuk urusan multimedia yang bersifat *open source* atau bebas dalam penggunaanya dan dapat berjalan dimana saja mulai dari komputer personal hingga *embedded device* atau gawai tanam bahkan ponsel pintar. *Pure Data* merupakan salah satu penghasil prosedural *audio*, yang memiliki artian yaitu proses pembuatannya dilakukan secara langsung (*real-time*) berdasarkan kumpulan aturan pemrograman dan sebuah masukan (*input*) (Farnell, 2007). Proses harus diaplikasikan dengan interval waktu yang menjamin bahwa sinyal latensi dari masukan ke keluaran tidak dapat diterima atau dirasakan oleh pendengaran manusia (Ausin et al., 2017).

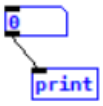
Prosedural *audio* sendiri adalah bentuk non-linear atau sistem yang tidak memiliki ketetapan, seringkali penerapannya ke arah jenis suara synthetis, dan dibuat secara langsung atau *real-time* sesuai dengan aturan pemrograman yang ada. Dapat dikatakan bahwa

prosedural *audio* adalah *computer sound*. Prosedural *audio* secara istilah merupakan sistem yang berfungsi dalam waktu nyata (*real-time*). Sifat dari prosedural *audio* tidak dihasilkan dari proses perekaman, dimana program dijalankan, biasanya dengan beberapa parameter untuk menentukan jenis keluaran yang diinginkan, dan secara langsung memulai menghasilkan suara dan musik yang berubah secara terus-menerus sepanjang waktu, atau merespon masukan dengan membuat suara dan musik (Farnell, 2007).

Pure Data merupakan percabangan dari bahasa pemrograman *patcher* yang diketahui sebagai *Max*. Pendistribusian utamanya dikembangkan oleh Miller Puckette. *Pure Data* didistribusikan dengan cara diubah ke dalam bentuk GUI serta diperbanyak ekternal *library* yang termasuk di dalamnya. *Pure Data* dapat membuat musisi, artis visual, seorang penampil, peneliti, hingga pengembang untuk menciptakan perangkat lunak secara *visual* atau gambar, tidak dengan menulis sekumpulan baris kode. *Pure Data* dapat digunakan untuk memproses dan menghasilkan suara, *video*, grafis dua dimensi/tiga dimensi, dan sensor antarmuka, masukan atau *input* dari sebuah alat atau *device*, dan MIDI (*Musical Intrument Digital Interface*). Di dalam *Pure Data* tersendiri, kita dapat menciptakan suara *synthesis* yang terkustomisasi, efek suara, pola musikal, hingga mesin musikal dengan mengkoneksikan obyek-obyek yang terdapat di dalam layar. *Pure Data* memiliki kemudahan dalam mempelajari kustomisasi sebuah musik dan pengembangan dalam ranah suara.

2.2. *Pure Data Patch*

Dokumen yang telah dibuat dari *Pure Data* dinamakan *patch*. Setiap dokumen memiliki satu jendela utama dan beberapa sub jendela. Sub jendela tersendiri dapat dibuka dan ditutup tetapi akan selalu berjalan apakah terlihat ataupun tidak terlihat. Pada Gambar 2.1 merupakan contoh sederhana sebuah *Pure Data patch*. Terdapat dua *box* atau kotak di dalam *patch* tersebut. Kotak *number* dan kotak *object* terkoneksi satu sama lain, keluaran dari kotak *number* terhubung ke masukan kotak *object*. Kotak ini memungkinkan untuk tidak memiliki atau bahkan memiliki banyak masukan maupun keluaran. Letak masukan atau *input* terdapat diatas kotak sedangkan keluaran atau *output* terletak dibawah kotak.



Gambar 2.1 *Pure Data patch* sederhana

2.3. *Pure Data* dengan *Heavy*

Heavy merupakan kerangka kerja atau *framework* untuk memudahkan dalam menghasilkan *plugin audio* atau fungsionalitas tambahan sebuah *audio*. Contoh penggunaan dari *plugin audio* tersebut adalah suara interaktif dan aplikasi musik seperti permainan, hingga instrumen musik tertentu. *Heavy* memanfaatkan prinsip perangkat lunak modern untuk menghasilkan kode C/C++ yang sangat teroptimasi, secara spesifik menyasar jenis-jenis populer arsitektur perangkat keras dan kerangka kerja sebuah perangkat lunak seperti *Unity*. *Heavy* dapat menginterpretasikan dan mengubah sekumpulan fitur dari *Pure Data patch*. Heavy tidak dapat memanfaatkan seluruh penggunaan kode *Pure Data* dan sepenuhnya tidak terikat oleh *Pure Data engine*.

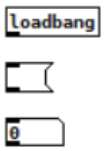
3. METODOLOGI

Metodologi pada penelitian ini menggunakan tata cara tertentu dalam menyajikan pembahasan *Pure Data*. Terdapat beberapa tahapan dalam menyelesaikan penelitian. Studi literatur digunakan sebagai teori penguat dan landasan dasar. Pada teori pendukung didapatkan dari jurnal, buku dan internet. Langkah selanjutnya adalah perancangan, yaitu menjelaskan bagaimana perancangan diimplementasikan dan bagian mana pada studi literatur yang diterapkan untuk *Real-time Audio Synthesis*. Kemudian, peneliti akan melakukan implementasi. Implementasi *Pure Data ke Unity* memanfaatkan *Heavy* sebagai penghasil skrip *audio*. Tahap selanjutnya adalah pengujian hasil implementasi. Pengujian menggunakan beberapa kasus uji diantaranya, melakukan pengujian unit dan validasi.

4. PERANCANGAN

Pada tahap ini akan dijelaskan bagaimana perancangan semua aset suara. Perancangan dilakukan dengan memanfaatkan bahasa pemrograman visual *Pure Data*. Dokumen yang dihasilkan dari bahasa pemrograman tersebut

adalah *Pure Data patch* atau *patch*. Dari perancangan ini menghasilkan dua jenis *patch* yaitu efek suara dan musik latar. Musik latar terbagi menjadi dua yaitu suara angin dan menu utama. Semua jenis suara hasil dari perancangan akan diimplementasikan ke dalam *Unity* melalui bantuan kerangka kerja *Heavy* yaitu menjembatani antara *Pure Data patch* ke dalam *Unity*.



Gambar 4.1 Jenis kotak pada *Pure Data patch*

Di dalam *Pure data patch* atau *patch* terdapat obyek-obyek yang dapat dihubungkan satu sama lain, dan memiliki fungsi yang berbeda-beda dalam pemogramannya. Gambar 4.1 merupakan beberapa contoh kotak atau *box*. Dimulai dari atas yaitu kotak *object*, *message*, dan *number*.

4.1. *Patch* Efek Suara

Pada permainan *platformer* di dalam penelitian ini efek suara akan hadir pada *scene* permainan. Yaitu pada sesi ketika karakter mengambil benda koleksi atau *collectible item*. Setiap benda koleksi yang didapatkan akan menghasilkan efek suara yang berbeda.

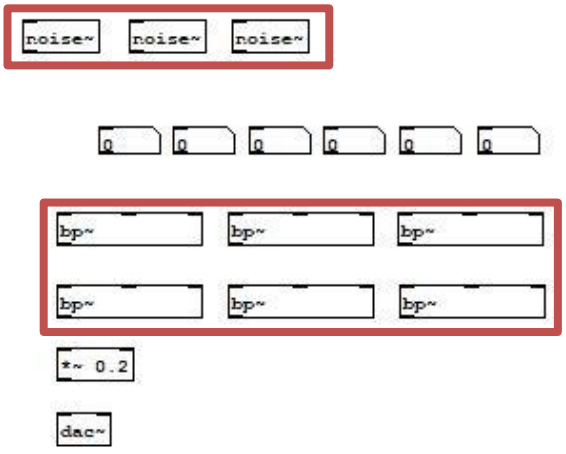
Gambar 4.2 Perancangan *patch* Efek Suara

Gambar 4.2 merupakan salah satu bagian dalam perancangan *patch* efek suara. Diawali dengan pembuatan delapan titik *sequencer* sederhana yang mana *sequencer* merupakan alat perangkat keras atau perangkat lunak yang dapat merekam, menerima atau mengirim, mengolah bunyi digital dari suatu instrumen musik elektronik (Widodo, 2006) yang berada pada penjelas warna oranye. Terdapat 8 kotak *number* dan kotak *object* bernama "*select*". Delapan kotak *number* ini berfungsi sebagai penyimpan frekuensi nada. Frekuensi nada didapatkan melalui referensi pada Gambar 2.13. Kotak *object* bernama "*select*" memiliki delapan argumen mulai dari angka 0 sampai 7. Delapan argumen akan dijalankan berurutan sesuai dengan tempo yang telah diprogram.

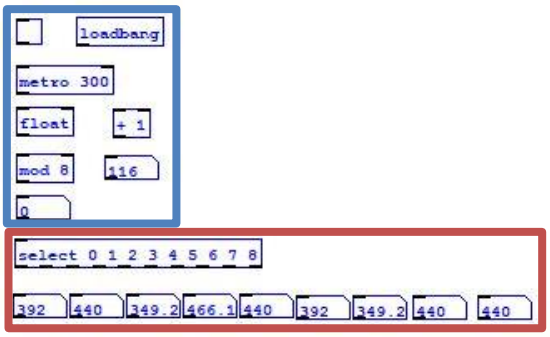
4.1. *Patch* Efek Suara

Musik latar (*background music*) di dalam permainan yang dibuat pada penelitian kali ini adalah suara angin dan menu utama. Pada saat game dimainkan musik latar suara angin akan terdengar tidak terlalu keras *volume* nya dan berjalan hingga permainan telah selesai, sedangkan musik latar menu dimainkan ketika pemain berada di menu utama.

Pada Gambar 4.3 merupakan salah satu bagian perancangan musik latar suara angin. Tahap ini membuat tiga buah *noise generator* yaitu kotak *object* bernama "*noise~*" dan *bandpass filter* yaitu kotak *object* bernama "*bp~*". Dengan menggabungkan keduanya dapat lebih mengontrol 'faktor-Q' berkaitan dengan hasil frekuensi. *Noise generator* berfungsi menghasilkan suara *noise*, sedangkan *bandpass filter* untuk menyaring suara *noise*. Terdapat enam buah *bandpass filter* bertujuan menghasilkan efek kedalaman suara.



Gambar 4.3 Perancangan *patch* Musik Latar Suara Angin



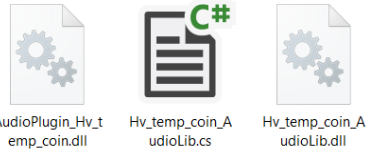
Gambar 4.4 Perancangan *patch* Musik Latar Menu Utama

Gambar 4.4 merupakan salah satu bagian perancangan musik latar menu utama. Tahap Penjelas warna oranye, merupakan tahap awal yang dilakukan yaitu membuat sembilan titik sequencer dengan memanfaatkan kotak *object* bernama "*select*", dan kotak *number*. Seperti pembahasan sebelumnya pada efek suara, "*select*" berfungsi untuk menjalankan kesembilan argumen mulai dari 0 sampai 8 secara berurutan. Kotak *number* akan menampilkan frekuensi setiap nada yang dipilih dan membuat alunan musik sesuai dengan nada yang disimpan.

Selanjutnya pada penjelas warna biru, pembuatan mesin metronom sama dengan perancangan efek suara, yaitu sebagai pengatur *tempo* yang diperlukan untuk kecepatan operasi matematika. Fungsi dari kotak *object* bernama "*float*", "*mod*", dan "+" sebagai operasi matematika. Hasil dari operasi itu ditampilkan dan terhubung dengan argumen "*select*" pada penjelas warna oranye. Kecepatan operasi matematika mempengaruhi kecepatan "*select*" dalam memilih argumen, sehingga alunan musik juga ikut terpengaruh.

5. IMPLEMENTASI

Pada pembahasan kali ini akan dijelaskan implementasi dari *Heavy* yang merupakan kerangka kerja untuk menghasilkan *audio* berupa skrip dalam Bahasa program C#. Komponen yang berada di dalam skrip tersebut berasal dari perancangan *Pure Data* sebelumnya. Sehingga dapat dikatakan *Heavy* menghubungkan *Pure Data* ke *Unity*. Hasil yang diperoleh dari *Heavy* selanjutnya akan dimanfaatkan sebagai *plugin audio* pada permainan *platformer* yang telah dibuat.



Gambar 5.1 Berkas-berkas hasil kerangka kerja *Heavy*

Mulai dari *Pure Data patch* yang merupakan bentuk perancangan *audio*, dilanjutkan dengan implementasi ke dalam *Heavy*. Skrip *audio* efek suara dan musik latar yang terbentuk melalui tahap modifikasi pada sebagian kode dengan tujuan tertentu dan untuk dapat diimplementasikan ke dalam *Unity*.

6. PENGUJIAN

Dalam bab ini membahas mengenai pengujian unit dan validasi. Pada pengujian unit kasus peneliti menguji algoritma dari skrip *audio* permainan pada Gambar 5.1. Pengujian menentukan *valid* atau tidak *valid* nya algoritma program yang diuji. Pengujian kedua adalah validasi. Terdapat beberapa kasus uji yang mencakup keseluruhan implementasi *Pure Data* ke dalam permainan.

7. KESIMPULAN DAN SARAN

Implementasi dari pemrograman visual *Pure Data* ke dalam permainan *platformer* telah berhasil dicapai, dan dapat dibuktikan dengan menghasilkan *audio* berupa efek suara, dan musik latar. *Audio* tersebut dihasilkan berdasarkan perancangan *Pure Data patch* yaitu dokumen keluaran *Pure Data*. Dengan ini dapat disimpulkan dari rumusan masalah, perancangan, implementasi, serta pengujian sebagai berikut :

1. Pemrograman visual *Pure Data* merupakan penghasil prosedural *audio* atau penghasil *audio* secara *real-time* yang memiliki jenis suara *synthesis*, dan dapat diimplementasikan ke dalam pengembangan *video game*. Berbeda dengan jenis teknologi *audio* tradisional lainnya, seperti rekaman suara *mp3* yang membutuhkan proses menangkap sinyal dengan menggunakan mikrofon, *mixing* suara hingga proses *mastering* untuk menjadi bentuk akhir dari *audio* tersebut.
2. Semua ukuran yang dihasilkan dari ketiga jenis suara dalam format *audio mp3*, lebih besar dibandingkan dengan skrip *audio*

yang merupakan hasil dari implementasi *Pure Data* ke *Heavy*. Selain itu dengan memanfaatkan skrip *audio* dapat memangkas jumlah dokumen yang digunakan dalam pengembangan *video game*. Dapat disimpulkan bahwa penggunaan *Pure Data* lebih efisien dibandingkan dengan *audio mp3*.

3. Memanfaatkan kerangka kerja alat bantu *Heavy*, merupakan salah satu cara untuk mengimplementasikan *Pure Data* ke dalam *Unity* dengan mudah. Hanya dengan mengakses laman kerangka kerja *Heavy* pengembang dapat menghasilkan skrip *audio* dan secara langsung digunakan.
4. Dilakukan dua pengujian fungsionalitas untuk menguji *Pure Data* yang sudah diimplementasikan ke dalam permainan, yaitu pengujian unit dan validasi. Dari setiap pengujian terdapat kasus uji yang menjadi tolak ukur keberhasilan dari pengujian. Hasil yang didapatkan dari setiap pengujian adalah status *valid* pada setiap kasus ujinya.

Berikut merupakan beberapa saran untuk implementasi lanjut dari pemrograman visual *Pure Data* :

1. Dalam penelitian kali ini haya menggunakan dan memanfaatkan kerangka kerja Heavy. Terdapat cara lain untuk mengimplementasikan pemrograman visual *Pure Data*, yaitu dengan memanfaatkan *library* bernama *Libpd*. Pemrograman visual *Pure Data* dapat diimplementasikan ke dalam aplikasi ponser pintar maupun *platform* pc *Windows* menggunakan *Libpd*.
2. Terdapat *Pure Data Reference Card* yaitu referensi kartu-kartu pada *Pure Data* karangan Karim BARKATI (2010) yang dapat diterapkan. Dikarenakan masih banyaknya kartu-kartu *Pure Data* pada penelitian ini yang belum diimplementasikan dan dimanfaatkan. Selain musik dengan suara berjenis *synthesis* terdapat banyak efek suara maupun musik latar yang dapat dibuat dan dikembangkan sesuai dengan *Pure Data Reference Card*.

8. DAFTAR PUSTAKA

Ausin, D.S., Pezzarossa, L. dan Schoeberl, M., 2017. *Real-Time Audio Processing on the T-CREST Multicore Platform*. Denmark: University of Denmark.

Blesinzki, C. 2000. *The Art and Science of Level Design*. USA: Game Developers Conference 2000.

Byrne, E., 2004. *Game Level Design*. USA: Charles River Media.

Farnell, A., 2007. *An introduction to procedural audio and its application in computer games*.

Gamasutra, 2016. *16-step Sequencer in Pure Data*. [online] Tersedia di: <https://www.gamasutra.com/blogs/AdamSporka/20161227/288411/16step_Sequencer_in_Pure_Data.php> [Diakses 6 Juni 2018]

Gormanley, S., 2013. *Audio immersion in games - a case study using an online game with background music and sounds effects*. United Kingdom: University of the West of Scotland.

Grimshaw, M., 2011. *Game Sound Technology and Player Interaction* : *Concepts and Developments*. United Kingdom : University of Bolton.

Hodgson, J., 2010. *Understanding Records*. United Kingdom : Bloomsburry Publishing.

Lindu, M., 2017. Pengembangan *Game Platformer* 2D Menggunakan Teknik *Projection Mapping*. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer. Malang : Universitas Brawijaya.

Minkkinen, T., 2016. *Basics of Platform Games*. Belanda: University of Applied Sciences.

Novation, 2018. *Launch Control*. [image online] Tersedia di: <https://us.novationmusic.com/launch/launch-control#> [Diakses 6 Juni 2018]

Peerdeman, P., 2010. *Sound and Music in Games*. Belanda: VU Amsterdam.

Pure Data, 2006. *Pd Documentation*. [image online] Tersedia di: <https://puredata.info/docs/manuals/pd/x2.htm#s1> [Diakses 6 Juni 2018]

Russ, M., 2009. *Sound Synthesis and Sampling*

Third Edition. Oxford: Elsevier Ltd..

Smith, G., Cha, M. dan Whitehead, J., 2008. *A Framework fr Analysis of 2D Platformer Levels*. California: University of California.

Widodo, T.W., 2006. Komputer dan Pengetahuan Program Aplikasi Musik Komputer. Yogyakarta: Institut Seni Indonesia Yogyakarta.

Wilcox, D., 2016. *PdParty: An iOS Computer Music Platform using libpd*. New York: Pure Data Conference.

Wei, T. dan Zheng, H., 2011. *Sound Effect of Physical Engine in Game Design*. Nanjing, Jiangsu, China: Xiaozhuang University.