การพัฒนาระบบอาษาแอสเซทแมเนจเมนบนแพลทฟอร์มอาษาเฟรมเวิร์ค (AAM)

A Development ARSA Assets Management on ARSA Framework platform (AAM).

อาษา ตั้งจิตสมคิด[1]

บทคัดย่อ

การวิจัยการพัฒนาระบบอาษาแอสเซทแมเนจเมนบนแพลทฟอร์มอาษาเฟรมเวิร์คหรือ AAM ในครั้งนี้มีวัตถุประสงค์เพื่อพัฒนาระบบ AAM บนแพลทฟอร์มอาษาเฟรมเวิร์ค และหาประสิทธิภาพของระบบ AAM ด้านความเร็วและความถูกต้องในการโหลดข้อมูลเข้าสู่เกมพบว่าระบบ AAM สามารถจัดการไฟล์ได้หลากหลายรูปแบบ ทั้งจาก Secondary Storage หรือ Cloud และการหาประสิทธิภาพด้านความเร็วของระบบ AAM ได้ทำการทดลองโหลดตัวละครชื่อ Mu จากเกม Zaros X Battle โดยมี Textures ทั้งหมด 307 ไฟล์, Audios ทั้งหมด 24 ไฟล์และ Text ไฟล์อื่นๆ 28 ไฟล์พบว่าประเภทไฟล์ที่มีประสิทธิภาพโหลดเร็วที่สุดคือ Pak+Zip+Pw มีค่าเฉลี่ยบนทุกอุปกรณ์ในการโหลดไฟล์แบบ actx (xml) ค่าเฉลี่ยอยู่ที่ 2.27 วินาที และแบบ act (text) ค่าเฉลี่ยอยู่ที่ 3.31 วินาที ส่วนการหาประสิทธิภาพด้านความถูกต้องของระบบ AAM ไฟล์ที่ถูกโหลดมาจาก Secondary Storage และ Cloud ยังคงทำงานได้ถูกต้อง ส่วนความเร็วในการโหลดไฟล์จาก Cloud จะช้าหรือเร็วนั้นปัจจัยที่มีผลกระทบน่าจะมาจากอัตราการรับและส่งข้อมูลของทั้งฝั่ง Server และ Client เป็นสาเหตุหลัก

คำสำคัญ อาษาเฟรมเวิร์ค อาษาแอสเซทแมเนจเมน ซารอสเอ็กแบทเทิล มิว

[1] ผู้ช่วยศาสตราจารย์ประจำวิทยาลัยนวัตกรรม มหาวิทยาลัยธรรมศาสตร์

Abstract

The objective of this research was to develop ARSA Assets Management (AAM) and to find out game loading speed and data accuracy performance. It was found that AAM can be handle varieties files from secondary storage and cloud. The result in-game loading speed: the author was testing on Mu character from Zaros X Battle that have 307 textures, 24 audios, 28 text files, it was found fastest file type in loading speed section is Pak+Zip+Pw, average speed both actx (xml) and act (text) are 2.27 and 3.31 seconds and the result into data accuracy performance: it was found files from both secondary storage and cloud work fine both loading into games and downloading from cloud. Upload and download speed depends on server and client speed ratio.

Keyword ARSA Framework, ARSA Asset Managerment AAM, Zaros X Battle Mu

บทนำ

ARSA Framework (2017) คือเครื่องมือสำหรับพัฒนาเกม ที่มีส่วนประกอบต่างๆ มากมายเช่น Rendering (Z. Ali and M. Usman, 2016: 119-1207), Physics, Audio/Video, Input (J. Munro, C. Boldyreff and A. Capiluppi, 2009) และ (S. H. M. Hashim and N. Mat Diah, 2015: 18-21) เป็นต้น โดยในงานวิจัยครั้งนี้จะขอเลือกพัฒนาระบบ Assets Management เพื่อนำไปใช้งานกับ ARSA Framework โดยระบบนี้มีหน้าที่เป็นตัวกลางระหว่างโปรแกรมเกม กับไฟล์ที่อยู่บน Secondary Storage เพื่อโหลดเข้าไปสู่หน่วยความจำ ความน่าสนใจของระบบนี้คือความเร็วและความถูกต้องในการโหลดข้อมูล โดยในเกมทั่วไปเมื่อมี Assets ขนาดใหญ่และต้องการจะโหลดใช้งานในเกม ระบบ Assets Management จะต้องทำหน้าที่รักษา

ความเร็วในการโหลดข้อมูล ให้เร็วที่สุดและถูกต้องที่สุดเท่าที่ระบบจะทำได้ไม่ว่าข้อมูลจะมาจากแหล่งใด ด้วยเหตุผลดังกล่าวผู้วิจัยจึงของเลือกพัฒนาระบบ ARSA Assets Management หรือ AAM (ต่อไปจะเรียก AAM) และทำการหาประสิทธิภาพด้านความเร็วและความถูกต้องของข้อมูลที่เข้ามาในระบบ AAM

วัตถุประสงค์

1. เพื่อพัฒนาระบบ AAM บนแพลทฟอร์มอาษาเฟรมเวิร์ค
2. เพื่อหาประสิทธิภาพของระบบ AAM ด้านความเร็วและความถูกต้องในการโหลดข้อมูลเข้าสู่เกม

วรรณกรรมที่เกียวข้อง

1. ARSA Framework

ภาพ 1. ARSA Framework คือเครื่องมือสร้างเกม 2 มิติและ 3 มิติ (S. Nykl, C. Mourning, M. Leitch, D. Chelberg, T. Franklin and Chang Liu, 2008: F3B-21-F3B-25) และ (Z. Pan Z G, He G Q, Yang B, 2008: 273–280) แบบ Open Source และ Cross Platform ที่ถูกพัฒนาโดย นายอาษา (ARSA Framework, 2017) จะถูกใช้เป็น Framework หลักในการพัฒนาระบบ AAM ในครั้งนี้

ภาพ 1 ARSA Framework logo

2. Zaros X Battle (Zaros X Battle, 2007)

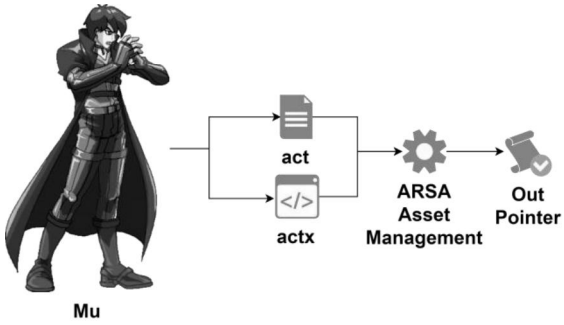
ภาพ 2. และ ภาพ 3. Zaros X Battle (M. P. Neto and J. R. F. Brega, 2015: 61-70) คือเกมต่อสู้ 2 มิติถูกวางตลาดเมื่อปี 2007 บนแพลทฟอร์ม PC และ Arcade ในปีนี้เกมจะถูก Re-Master อีกทั้งบนแพลทฟอร์ม Steamและ Android

ภาพ 2. Zaros X Battle Cover PC Version

ภาพ 3. Zaros X Battle In-game screenshot

ทางผู้วิจัยจึงเลือกใช้เกมนี้ เพื่อเพื่อหาประสิทธิภาพ ด้านความเร็วและความถูกต้องในการโหลดข้อมูลเข้าเกม โดยได้เลือกตัวละครจากเกมคือพระเอกชื่อ Mu ตาม ภาพ 4. โดยมีไฟล์ Textures ทั้งหมด 307 ไฟล์, Audios ทั้งหมด 24 ไฟล์และ Text ไฟล์อื่นๆ 28 ไฟล์ เมื่อทำการศึกษาตัวละคร Mu พบว่าข้อมูลของตัวละครนั้นถูกเก็บไว้ 2 รูปแบบคือ act และ actx เมื่อทำการศึกษาจึงทราบว่า .act คือการอ่าน

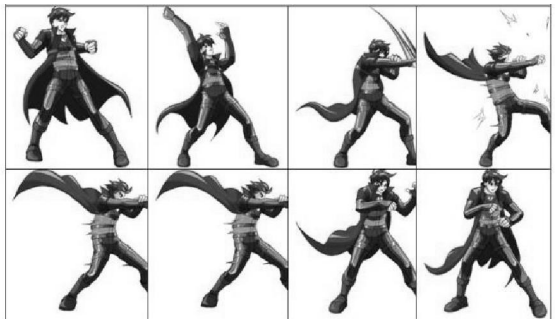
ข้อมูลจาก text โดยตรง ส่วน actx (Jingming Xie, 2011: 635-639) จะอ่านจากไฟล์ xml เพราะฉะนั้นในการหาประสิทธิภาพด้านความเร็วในการโหลดข้อมูลครั้งนี้จึงต้องทำการทดลองกับทั้ง 2 ไฟล์



ภาพ 4. Mu Character pipeline

3. Textures

ภาพ 5. ไฟล์ภาพ 2 มิติแบบราสเตอร์มีหลากลายรูปแบบเช่น jpg, png, pcx, bmp, tga หรือ psd เป็นต้น ไฟล์เหล่านี้จะถูกใช้สร้างขึ้นและร้อยเรียงกันเป็นภาพแอนิเมชันเพื่อสร้างขึ้นมาเป็นตัาละครในเกม



ภาพ 5. Mu Character

4. Audios/ Videos

ไฟล์เสียงและวีดีโอใช้แสดงประกอบเป็นเสียงพากษ์ของตัวละครมีหลากหลายรูปแบบเช่น wav, mp3, mp4, ogg, mov, mpg, mpg2, avi, wma และ wmv เป็นต้น

5. การบีบอัดข้อมูล

ใช้บีบอัดข้อมูลของ Texture, Audio, Videos และไฟล์อื่นๆเพื่อรวมเป็นไฟล์หลักหรือให้ขนาดลดลง จากนั้นเมื่อต้องการจะใช้งานจึงทำการถอดรหัสข้อมูลแล้วนำไปประมวลผลเป็นลำดับต่อไป การบีบอัดข้อมูลในปัจจุบันมีมากมายหลายรูปแบบ ซึ่งในงานวิจัยครั้งนี้จะสนใจ 3 รูปแบบคือ 1.) PAK คือการบีบอัดข้อมูลดิบโดยตรง 2.) Zip เป็นการบีบอัดโดยใช้อัลกอริทึมแบบ LZMA สามารถเข้ารหัสไฟล์โดยใช้อัลกอริทึม AES256 เพื่อปกป้องข้อมูลได้อีกชั้นหนึ่ง 3.) Tar คือการนำไฟล์มารวมกันเป็นไฟล์เดียวโดยไม่มีการบีบอีดข้อมูลผู้วิจัยจะใช้ไฟล์ 3 ประเภทนี้มาเพื่อหาประสิทธิภาพความเร็วและความถูกต้องในการโหลดข้อมูลทั้งจาก Secondary Storage หรือ Cloud

วิธีดำเนินการวิจัย

1. การออกแบบระบบ AAM

1.1 ไฟล์ที่สำคัญของระบบ AAM คือ 1.) Textures 2.) Audios 3.) Videos และ 4.) Pak,Zip,Tar ซึ่งทั้ง 4 ประเภทอาจมาได้จากการอ่าน Secondary Storage หรือโหลดจาก Cloud

1.2 เมื่อระบบได้รับทราบถึงไฟล์แล้วต้องทำการคัดกรองหากเป็นไฟล์ประเภทที่ 1 ถึง 3 ระบบจะถือว่าเป็นการอ่านไฟล์แบบ Directly โดยระบบจะทำการคัดแยกอีกชั้นว่าเป็นการโหลดไฟล์เข้าระบบหรือจะเป็นการสร้าง Pak ไฟล์โดยมีเงื่อนไขดังนี้

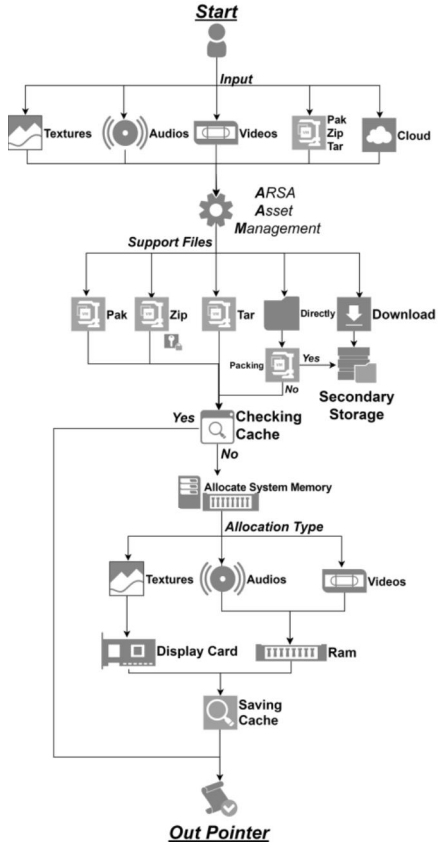
A.) หากเป็นการสร้าง Pak จะทำการรวมไฟล์ทั้งหมดเข้าไว้ด้วยกันและทำการเขียนลง Secondary Storage

B.) หากเป็นการโหลดเข้าระบบจะมีการคัดกรองในขั้นต่อไปกล่าวคือหากมีไฟล์ชื่อนี้อยู่แล้วจะส่งคืนค่า Pointer กลับไปทันที แต่หากไม่มีระบบย้ายไฟล์จาก Secondary Storage ไปสู่หน่วยความจำจากนั้นคัดกรองหากไฟล์นี้เป็น Texures

จะย้ายความจำจากระบบเข้าสู่หน่วยความจำของ Display Card แต่หากเป็นไฟล์ Audios หรือ Videos จะยังคงหน่วยความจำนี้ไว้ที่เดิม

C.) ทำการบันทึกชื่อไฟล์นี้เพื่อใช้เป็นคีย์หลักในการค้นหาต่อไป

1.3 ไฟล์ประเภทที่ 4.) หากเป็น Pak หรือ Tar จะทำการจำแนกไฟล์ที่รวมกันจากข้อ A แล้วทำการโหลดโดยใช้กระบวนการเดียวกับข้อ B แต่หากเป็นไฟล์ Zip จะมีการจำแนกเป็นแบบมีและไม่มีรหัสผ่าน เพื่อนำไปขยายไฟล์ที่บีบอัดไว้แล้วนำเข้าระบบตามข้อ B.) และสุดท้ายทุกไฟล์ต้องทำการบันทึกตามข้อ C เช่นเดียวกัน ดังภาพ ภาพ 6.



ภาพ 6 AAM Pipeline

2 การทดลองหาประสิทธิภาพของระบบ AAM

ในการทดสอบระบบ AAM ผู้วิจัยได้สนใจในเรื่องของความเร็ว ในการโหลดข้อมูลเข้าสู่ระบบ และความถูกต้องของข้อมูลเมื่อมีการโหลดไฟล์มาจาก Cloud จึงได้ทำการออกแบบการทดลองไว้ 2 ส่วนดังต่อไปนี้

2.1 การหาประสิทธิภาพด้านความเร็วของระบบ AAM

ในการโหลดไฟล์เข้าสู่ระบบ AAM จะทดลองกับตัวละครชื่อ Mu จากเกม Zaros X Battle โดยมีรายละเอียดการทดลองดังต่อไปนี้

A) มี Textures ทั้งหมด 307 ไฟล์แยกเป็นไฟล์ PCX 251 ไฟล์, PNG 55 ไฟล์ และ BMP 1 ไฟล์ แต่ละไฟล์มีขนาดและ Format ที่แตกต่างกันไป

B) มี Audios ทั้งหมด 24 ไฟล์เป็นไฟล์ประเภท WAV ทั้งหมด

C) มี Text ไฟล์อื่นๆทั้งหมด 28 ไฟล์ โดยจะแยกทำการทดลองสองครั้งระหว่างไฟล์ act (text) และไฟล์ actx (xml)

D) กำหนดไฟล์ตัวอย่างสำหรับทำการทดลองทั้งหมด 8 ตัวอย่างได้แก่

1. pak คือไฟล์ที่รวมไฟล์ย่อยทั้งหมดจากข้อ 3.2.1, 3.2.2 และ 3.2.3 ด้วยวิธี ARSA Pak

2. pak+zip คือนำไฟล์จากข้อ 1. มาบีบอัดด้วยวิธี LZMA

3. pak+zip+pw คือนำไฟล์จากข้อ 1. มาบีบอัดและเข้ารหัสด้วยวิธี AES256

4. pak+tar คือไฟล์ที่รวมไฟล์ย่อยทั้งหมดจากข้อ 3.2.1, 3.2.2 และ 3.2.3

5. directly คือโหลดไฟล์จาก Secondary Storage โดยตรง

6. direct+zip คือนำไฟล์จากข้อ 1. มาบีบอัดด้วยวิธี LZMA

7. direct+zip+pw คือนำไฟล์จากข้อ 1. มาบีบอัดและเข้ารหัสด้วยวิธี AES256

8. direct+tar คือไฟล์ที่รวมไฟล์ย่อยจากข้อ A.), B.) และ C.) ทั้งหมด

E.) ทำการทดลองระบบ AAM บนอุปกรณ์และระบบปฏิบัติการที่แตกต่างกันทั้งหมด 9 อุปกรณ์มีรายละเอียดตาม ตาราง 1

ตาราง 1 อุปกรณ์และระบบปฏิบัติการ

ลำดับ	อุปกรณ์	ระบบปฏิบัติการ
1.	Desktop	Windows 7
2.	Alcatel OneTouch D5	Android 4.4.2
3.	AIS Lava Iris 800	Android 4.4
4.	Dtac Joey Selfie 4.5	Android 4.4
5.	Samsung Galaxy A8	Android 5.1.1
6.	Samsung Galaxy Note 10.1	Android 4.0.3
7.	Lenovo A369i	Android 4.2.2
8.	Cherry Paladin Plus	Android 4.2.2
9.	Bluestack 2	Android (Emulator)

2.2 การหาประสิทธิภาพด้านความถูกต้องของระบบ AAM

ผู้วิจัยได้ทำการออกแบบ การทดลองด้านความถูกต้องของข้อมูล มีขั้นตอนดังต่อไปนี้

A) กำหนด Server ในการทดลองวัดความเร็วโหลดไฟล์จาก Cloud โดยใช้ Server ของ Google และ Dropbox มีไฟล์ทั้งหมด 3 ไฟล์ที่ใช้ในการทดลองได้แก่

1. Tar ไฟล์ที่รวมข้อมูล Textures, Audios และ Pak เข้าไว้ด้วยกันมีขนาด 59.1 Mbs

2. Zip ไฟล์ที่บีบอัดด้วยวิธี LZMA ของข้อมูล Textures, Audios และ Pak เข้าไว้ด้วยกันมีขนาด 56.7 Mbs

3. Zip+PW ไฟล์ที่บีบอัดและเข้ารหัสด้วยวิธี AES256 ของข้อมูล Textures, Audios และ Pak เข้าไว้ด้วยกันมีขนาด 56.3 Mbs

3. สรุปขั้นตอนวิธีดำเนินงาน

สรุปขั้นตอนวิธีการดำเนินงานของระบบ AAM ได้แก่ 3.1 การออกแบบระบบ AAM ให้สามารถรองรับไฟล์หลากหลายทั้งจากการอ่าน Secondary Storage และ Cloud และ 3.2 การทดลองหาประสิทธิภาพของระบบ AAM โดยแบ่งการทดลองออกเป็น 2 ส่วนคือ ด้านการวัดความเร็วและความถูกต้องในการโหลดข้อมูลจาก Cloud โดยนำตัวอย่างไปทดลองบน 9 อุปกรณ์ที่แตกต่างกัน

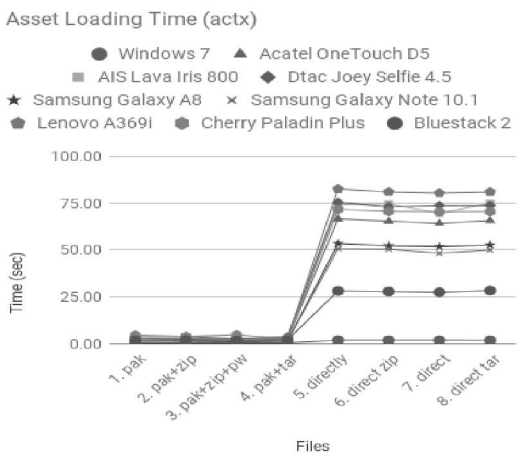
ผลการทดลอง

ผลการทดลองการหาประสิทธิภาพด้านความเร็วและความถูกต้องของข้อมูลในระบบ AAM มีดังต่อไปนี้

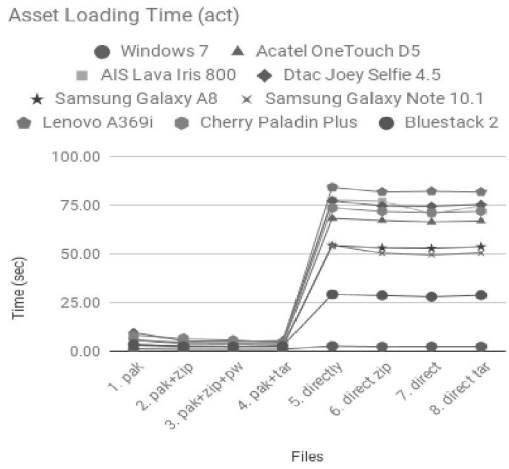
1. การหาประสิทธิภาพด้านความเร็วของระบบ AAM

จาก ภาพ 7. ถึง ภาพ 10. จะแสดงความเร็วในการโหลดไฟล์ act และ actx ทั้ง 8 ประเภทบน 9 อุปกรณ์ โดยไฟล์ที่โหลดเร็วที่สุดคือไฟล์ actx ประเภท 3. pak+zip+pw มีความเร็วเฉลี่ยอยู่ที่ 2.27 วินาที และไฟล์ที่โหลดช้าที่สุดคือไฟล์ act ประเภท 5. directly มีความเร็วเฉลี่ยอยู่ที่ 58.08 วินาที โดยผู้วิจัยได้ตั้งข้อสังเกตจากผลการทดลองนี้ไว้ว่าไฟล์ actx จะโหลดได้เร็วกว่าไฟล์ act เสมอในทุกประเภท

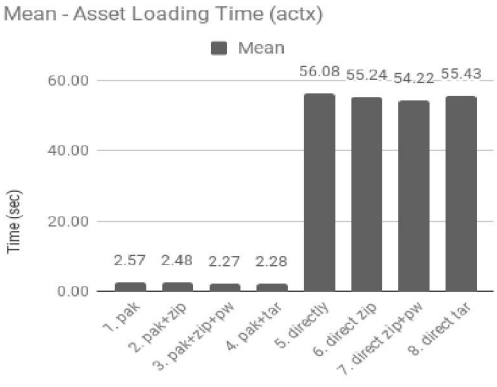
ส่วนประเภทไฟล์ที่ผ่านการ Pak, Zip หรือ Tar มาก่อนจะมีอัตราความเร็วในการโหลดได้เร็วกว่าไฟล์ที่ทำการอ่านโดยตรงหรือ directly เสมอโดยรายละเอียดเวลาในการโหลดของทุกไฟล์และทุกอุปกรณ์ทั้งในส่วนของ act และ actx แสดงใน ตาราง 2 และ ตาราง 3



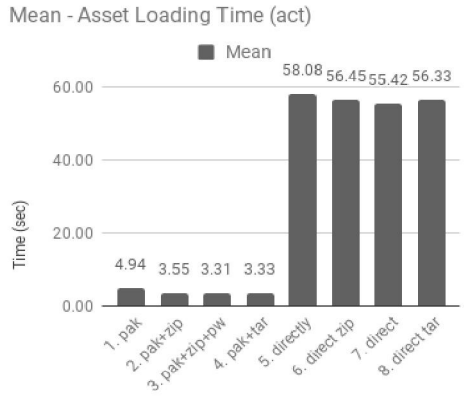
ภาพ 7 Asset Loading Time (actx)



ภาพ 8 Asset Loading Time (act)



ภาพ 9 Mean - Asset Loading Time (actx)



ภาพ 10 Mean - Asset Loading Time (actx)

ตาราง 2 Data - Asset Loading Time (actx)

Device/Files	1	2	3	4	5	6	7	8
Windows 7	0.63	0.63	0.60	0.64	2.03	1.99	1.99	1.97
Acatel OneTouch D5	2.74	2.51	2.12	2.51	66.56	65.39	64.24	65.68
AIS Lava Iris 800	2.85	2.90	2.68	2.86	74.13	74.58	69.58	74.91
Dtac Joey Selfie 4.5	3.96	3.92	2.80	3.81	75.43	73.00	73.70	73.70
Samsung Galaxy A8	1.50	1.46	1.21	1.46	53.47	52.35	51.92	52.77
Samsung Galaxy Note 10.1	1.75	1.73	1.52	1.62	50.68	50.47	48.28	50.02
Lenovo A369i	2.98	2.89	2.69	2.87	82.50	80.90	80.35	80.87
Cherry Paladin Plus	4.72	4.07	4.90	2.65	71.65	70.55	70.35	70.43
Bluestack 2	2.00	2.17	1.87	2.11	28.28	27.97	27.56	28.48
Mean	2.57	2.48	2.27	2.28	56.08	55.24	54.22	55.43

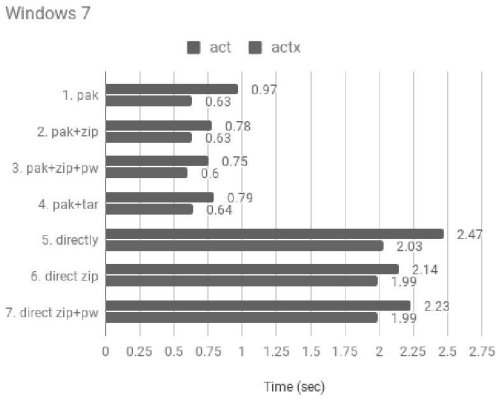
1. pak	2. pak+zip	3. pak+zip+pw	4. pak+tar
5. direc	6. direct zip	7. direct zip+pw	8. direct tar

ตาราง 3 Data - Asset Loading Time (act)

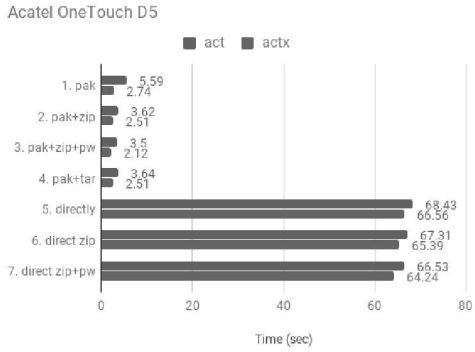
Device/Files	1	2	3	4	5	6	7	8
Windows 7	0.97	0.78	0.75	0.79	2.47	2.14	2.23	2.18
Acatel OneTouch D5	5.59	3.62	3.50	3.64	68.43	67.31	66.53	67.07
AIS Lava Iris 800	5.67	4.40	3.94	4.29	77.90	77.27	70.90	74.87
Dtac Joey Selfie 4.5	9.42	5.25	5.15	5.17	77.51	74.83	74.68	75.52
Samsung Galaxy A8	2.79	2.26	2.15	2.28	54.38	53.25	52.98	53.82
Samsung Galaxy Note 10.1	3.09	2.42	2.15	2.40	54.49	50.47	49.59	50.60
Lenovo A369i	5.65	4.35	4.21	4.36	84.47	82.13	82.45	82.09
Cherry Paladin Plus	8.00	6.50	5.61	4.41	73.89	72.03	71.47	72.10
Bluestack 2	3.29	2.40	2.30	2.62	29.15	28.60	27.93	28.71
Mean	4.94	3.55	3.31	3.33	58.08	56.45	55.42	56.33

1. pak	2. pak+zip	3. pak+zip+pw	4. pak+tar
5. direc	6. direct zip	7. direct zip+pw	8. direct tar

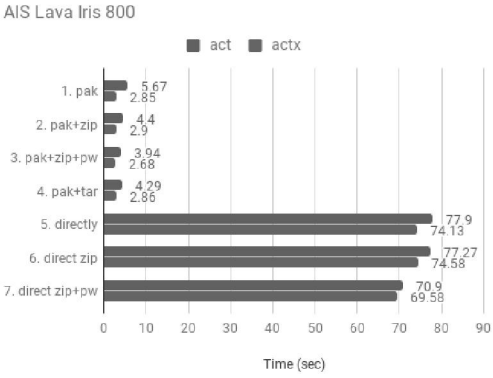
จากภาพ 11. ถึง ภาพ 19. จะเป็นการเปรียบเทียบความเร็วระหว่างไฟล์ act และ actx ในแต่ละอุปกรณ์เพื่อให้เห็นความชัดเจนถึงประสิทธิภาพได้จากระบบ AAM โดยผลลัพธ์ที่ได้บ่งบอกได้อย่างเอกฉันท์ว่าไฟล์ actx ประเภท pak+zip+pw โหลดได้เร็วที่สุดในทุกอุปกรณ์



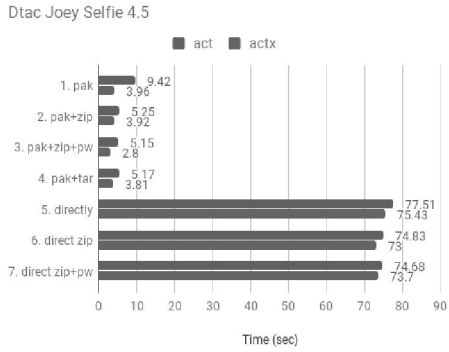
ภาพ 11 Windows7 - Loading Time



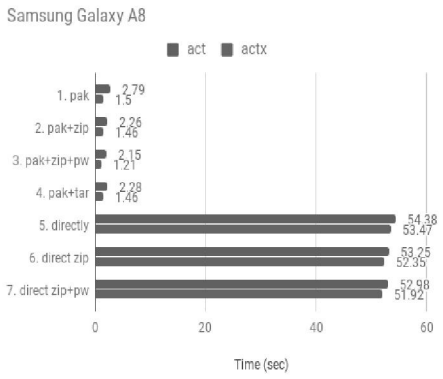
ภาพ 12 Acatel OneTouch D5 - Loading Time



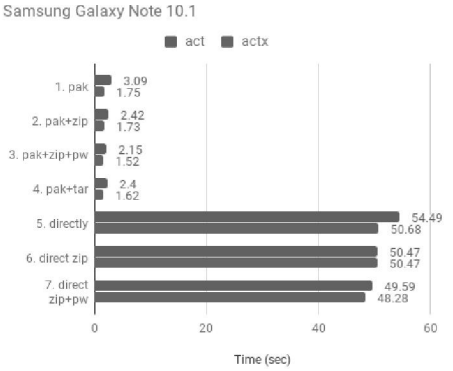
ภาพ 13 AIS Lava Iris 800 - Loading Time



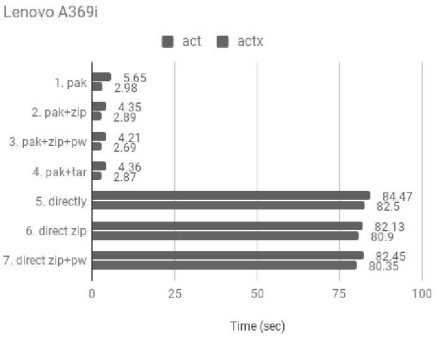
ภาพ 14 Dtac Joey Selfie 4.5 - Loading Time



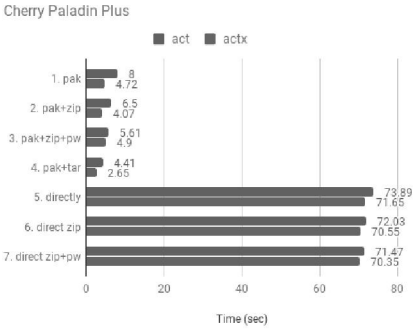
ภาพ 15 Galaxy A8 - Loading Time



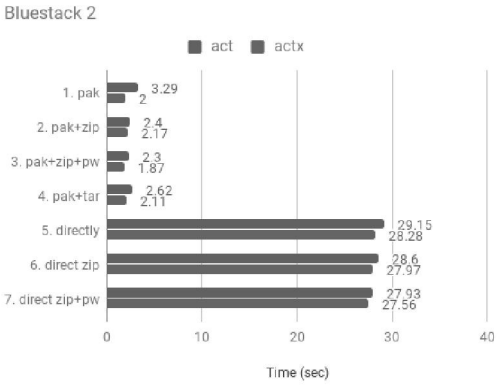
ภาพ 16 Galaxy Note 10.1 - Loading Time



ภาพ 17 Lenovo A369i - Loading Time



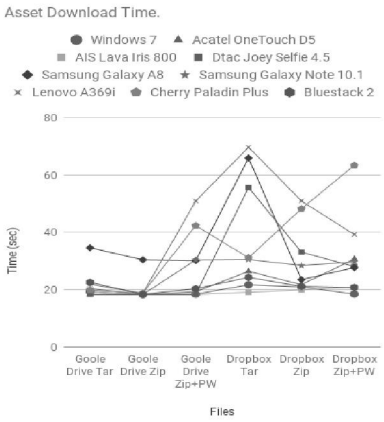
ภาพ 18 Cherry Paladin Plus - Loading Time



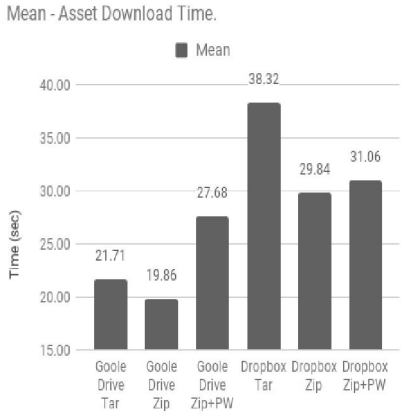
ภาพ 19 Bluestack 2 - Asset Loading Time

2. การหาประสิทธิภาพด้านความถูกต้องของระบบ AAM

จาก ภาพ 20. และ ภาพ 21. ผลลัพธ์จะสังเกตเห็นว่า ค่าเฉลี่ยของความเร็วที่วัดได้จากทุกอุปกรณ์ Dropbox ใช้เวลาโหลดนานที่สุดคือ 38.32 วินาที ส่วนไฟล์จาก Google ใช้เวลาในการโหลดที่น้อยกว่าคือ 19.86 วินาที แต่ทว่าปัจจัยที่สำคัญน่าจะมาจากอัตราความเร็วการรับและส่งข้อมูลของทั้งฝั่ง Server-Client ในขณะทำการทดลอง อีกทั้งขนาดของไฟล์ทั้งสาม ประเภทไม่ได้มีขนาดแตกต่างกันอย่างชัดเจน จึงไม่สมควรจะสรุปว่า ความเร็วที่มากหรือน้อย จะส่งผลกระทบต่อความถูกต้องของข้อมูลที่จะนำเข้าสู่ระบบ AAM เพราะข้อมูลที่ได้ในทุกกรณี มีความถูกต้องเหมือนกันทั้งหมด เมื่อโหลดไปใช้ในเกม Zaros X Battle



ภาพ 20 Asset Download Time



ภาพ 21 Mean - Asset Download Time

สรุปและอภิปรายผล

การวิจัยเรื่องการพัฒนาระบบอาษาแอสเซทแมเนจเมนบนแพลทฟอร์มอาษาเฟรมเวิร์ค (AAM) ในครั้งนี้มีวัตถุประสงค์เพื่อพัฒนาระบบ AAM และหาประสิทธิภาพของระบบ AAM ด้านความเร็วและความถูกต้องในการโหลดข้อมูลโดยตัวระบบ AAM ที่พัฒนาขึ้นนี้สามารถจัดการไฟล์ได้หลากหลายประเภท อย่างรวดเร็วและแม่นยำทั้งจาก Secondary Storage และ Cloud

การหาประสิทธิภาพ ด้านความเร็วของระบบ AAM ได้ทำการทดลองโหลดตัวละครชื่อ Mu จากเกม Zaros X Battle พบว่าประเภทไฟล์ที่มีประสิทธิภาพโหลดเร็วที่สุดคือ Pak+Zip+Pw มีค่าเฉลี่ยบนทุกอุปกรณ์ในการโหลดไฟล์แบบ actx (xml) อยู่ที่ 2.27 วินาที และแบบ act (text) อยู่ที่ 3.31 วินาที

ส่วนการหาประสิทธิภาพด้านความถูกต้องของระบบ AAM ไฟล์ที่ถูกโหลดมาจาก Secondary Storage และ Cloud ยังคงทำงานได้ถูกต้อง ส่วนความเร็วในการโหลดไฟล์จาก Cloud จะช้าหรือเร็วนั้นปัจจัยที่มีผลกระทบน่าจะมาจากอัตราการรับและส่งข้อมูลของทั้งฝั่ง Server และ Client เป็นสาเหตุหลัก

เอกสารอ้างอิง

ARSA Framework. (2017). **Sanook**. Retrieved April 7, 2017, from http: //game.sanook.com/985401

Z. Ali and M. Usman (2016). "A framework for game engine selection for gamification and serious games". **Future Technologies Conference (FTC,2016)**: 1199-1207.

J. Munro, C. Boldyreff and A. Capiluppi, (2009). "Architectural studies of games engines: The Quake series". **International IEEE Consumer Electronics Society's Games Innovations Conference (2009)**: 246-255.

S. H. M. Hashim and N. Mat Diah (2015). "Game engine framework for Non-Programming background," **IEEE Conference on Open Systems (ICOS, 2015)**: 18-21.

S. Nykl, C. Mourning, M. Leitch, D. Chelberg, T. Franklin and Chang Liu (2008). "An overview of the STEAMiE Educational game Engine".

38th Annual Frontiers in Education Conference 2008: F3B-21-F3B-25.

Z. Pan Z G, He G Q and Yang B, (2008). "Advance in research on mobile graphics technologies" **Journal Of Computer-aided Design & Computer Graphics.** 120 (3): 273–280

Zaros X Battle. (2007). **Manager Online**. Retrieved May 31, 2007, from: http: //www.manager.co.th/Game/ViewNews.aspx?NewsID= 9500000062739

M. P. Neto and J. R. F. Brega (2015). "A Survey of Solutions for Game Engines in the Development of Immersive Applications for Multi-projection Systems as Base for a Generic Solution Design". **XVII Symposium on Virtual and Augmented Reality 2015:** 61-70.

Jingming Xie (2011). "The research on mobile game engine". **International Conference on Image Analysis and Signal Processing 2011:** 635-639