**רישומי פיתוח**

התקנת Visual Studio

* הורדת גרסת Community
* לבחור באופציות להתקנה את שלושת האפשרויות תחת Windows
* לבחור תחת Gaming אופציה Gaming Development with C++
* מצד ימין לסמן מלבד C++ profiling tools אתת גרסת ה - Windows 10 SDK העדכנית ביותר ואת גרסת 8.1 SDK וגם את Unreal Engine Installer

יצירת VCS מקומי

* מזינים את כתובת התיקייה
* מוודאים שבחור Git
* לוחצים Create ב - VCS
* אם יש התראה שהתיקייה כבר קיימת, יש לאשר הקמת Repo בתיקייה קיימת
* יוצרים את הפרויקט
* בוחרים Create

יצירת gitignore.

* לחיצה ימנית על אחד התיקיות שאנחנו רוצים להתעלם מהם
* בוחרים Ignore
* מסמנים Ignore everything beneath ובוחרים את התיקיה הרצויה
* מאשרים
* כעת נוצר קובץ ה - .gitignore
* כעת ניתן להוסיף את כל התיקיות האוטומטיות ל - Ignore
  + Build
  + Binaries
  + Intermediate
  + DerivedDataCache
  + Saved
  + \*VC.db
  + \*.vs
* בקובץ הטקסט של  ה - Ignore יש לרשום אותם כך: Build/\*
* הערות ניתן להוסיף אחרי סולמיות #
* בוחרים Stage All
* ועושים Initial Commit

יצירת VCS GitHub

* בתוך GitHub לבחור New Repository
* להזין שם (בפועל זה שם התיקיה שתיווצר במחשב)
* להזין תיאור
* לבחור להתחיל את הפרויקט עם קובץ README
* לבחור Unreal ל - .gitignore
* לבחור רישיון מתאים וליצור את ה – Repository
* כעת ניתן לערוך את קובץ ה – README
* יש לבחור בקובץ ה – README
* מצד ימין לבחור בעט – משמעותו עריכה
* סימני עריכה:
  + # משמעותו כותרת ברמה 1
  + \*\*\*\* יוצר קו מפריד
  + ליצור קישור ע"י הכנסת הטקסט לסוגריים מרובעות ומיד לאחר מכן את הכתובת בסוגריים עגולות. לדוגמה: [here](Link address)
  + \* אחת עושה רשימה
* כאשר שומרים בעצם יוצרים Commit חדש
* ב – VCS בוחרים כלום ואז אפשרויות ואז Authentication
* ומזינים את חשבון ה – GitHub על מנת לייצר קישור ל - VCS
* ב – VCS בוחרים קובץ ואז Clone/New
* ובסרגל למעלה בצד שמאל ב - Remote
* יופיע רשימת ה – Repositories שלך מ – GitHub ומתחתיו בוחר ב – Clone
* בוחרים את כתובת הקובץ הרצויה (יש להשאיר את שם הפרויקט ולשנות את שארית הכתובת)
* פותחים את המנוע ובוחרים פרויקט חדש ויוצרים אותו בתיקיית ה – Repository (שם הפרויקט ייצור תיקיה חדשה נוספת פנימה ואין צורך להכין אותה ידנית)
* הערה: במידה ויוצרים את הפרויקט בתיקיה חדשה נוספת בתוך התיקיה שבו יש את ה – Repository, לכן יש צורך להעביר את ה - .gitignore לתוך התיקיה החדשה הנכונה
* חוזרים ל – VCS ועושים Stage רק לקובץ .gitignore הישן והחדש
* כעת ניתן לעשות Stage All
* מייצרים Initial Commit

אתחול פרויקט VCS

* בתוך SourceTree לעשות CheckOut או Reset להתחייבות עובדת
* למחוק את כל הקבצים האוטומטיים בנוסף קובץ ה - .sln
* לפתוח מחדש את המנוע מהפרויקט
* פתיחת קובץ ה - .sln החדש
* סגירת הפרויקט, לחיצה ימינית על קובץ הפרויקט של המנוע ובחירת:

Generate Visual Studio Project Files

שדרוג גרסת מנוע

* מורידים את המנוע החדש
* מוודאים שישנו Commit עדכני לפרויקט
* פותחים את המנוע החדש
* בוחרים את הפרויקט לשדרוג
* בוחרים Open
* ואז בוחרים More Options במקום בפתיחת העתק
* בוחרים Convert In Place
* בודקים שהכל עובד כרגיל
* שומרים ועושים Commit חדש
* בוחרים Tag
* מוסיפים שם של גרסת המנוע
* ובחורים Specified Commit ומאשרים

פרויקט חדש

* מכינים שלב חדש מבלי לשמור את הקודם
* מכניסים אותו תחת הגדרות פרויקט Maps and modes כברירת מחדל
* תחת תיאור בהגדרות הפרויקט ממלאים את ה – Copyright notice לפני שמתחילים לעבוד

Includes ב – Visual Studio

באנריל צריך להוסיף header מתאים למצביעים שאתה רוצה לגשת אליהם.

לדוגמא לקבלת GetOwner (ב – Actor Component) יש להוסיף:

#include "GameFramework/Actor.h"

בקבצי .cpp יש להוסיף מתחת ל – Include הראשוני, ובקובץ .h מעל ה – Include הראשוני

**פרויקט Building Escape**

* הכנת החדר שממנו רוצים לברוח
* הוספת Collision לדלתות שלא ניתן יהיה לעבור דרכן
* הכנת Default Pawn חדש ב – GameMode חדש
  + ניתן ל – Pawn אופציה Simulate Physics תחת CollisionComponent
  + מאסה כשבעים קילו
  + תחת Class Defaults גובה 60 ל – Base Eye Hight
  + על מנת לתקן מצב שה – Pawn לא יזוז ע"י חפצים חיצוניים תחת Physics ב – CollisionComponent יש לנעול רוטציה תחת Constraints ב - X Y Z
  + כמובן שיש לעדכן את ה – GameMode בהגדרות הפרויקט
* הפיכת החפצים שפותחים את הדלת ל – PhysicsBody ע"י Simulate Physics וקביעת מאסה, חשוב לא לשכוח לסמן Generate overlap Events אחרת ה – Trigger Volume לא ייתן תוצאה
* להוסיף C++ Component ל – Pawn בשם Grabber ו - PhysicsHandle

ארכיטקטורת Grabber

ב – BeginPlay נשמור לתוך שני מצביעים את ה – InputComponent ו – PhysicsHandle ע"י FindComponentByClass מתוך ה – GetOwner.

מיד לאחר מכן נבצע הגנה על הפוינטר של ה – PhysicsHandle (InputComponent יורד בתורשה ולכן אין צורך להגן עליו).

ב – BeginPlay ניתן כבר לבצע את ה – Action Binding מה – Input Component (כמובן לאחר שהגדרנו אותם ב – Input בהגדרות הפרויקט).

כעת נגדיר פונקציית Grab שקוראת לפונקציית RayTrace ומחזירה את ה – FHitResult.

אם נמצא Actor אז דרך ה – PhysicsHandle מבצעים GrabComponent.

כעת ב – Tick בודקים כל הזמן אם PhysicsHandle יש לו GrabbedComponent אין צורך בסגירת סוגריים אחרי GrabbedComponent. אם ישנו חפץ שמחזיקים אז כל שניה נגדיר מחדש את מיקומו ע"י SetTargetLocation מתוך ה – PhysicsHandle.

כאשר עוזבים את החפץ אז קוראים לפונקציית Release פשוטה שמשתמשת ב – PhysicsHandle לשחרר אותו.

ארכיטקטורת OpenDoor

נכריז על DECLARE\_DYNAMIC\_MULTICAST\_DELEGATE(FEventName);

בתחילת ה – Class מסוג C++ Actor Component שנלביש על הדלת ונהפוך אותו לסוג Blueprint.

נגדיר ב – Class שני Events אחד לפתיחה ואחד לסגירת הדלת.

לדוגמה:

UPROPERTY(BlueprintAssignable)

FDoorEvent FDoorOpen;

כעת ב – Blueprint נלחץ על ה – ActorComponent ורק אז נוכל לבחור את ה – Events ב – Blueprints.

* לחצן ימיני ברקע
* בחירת AddEvent
* Event Dispatchers ובחירת ה – Event הרצוי שהכנו

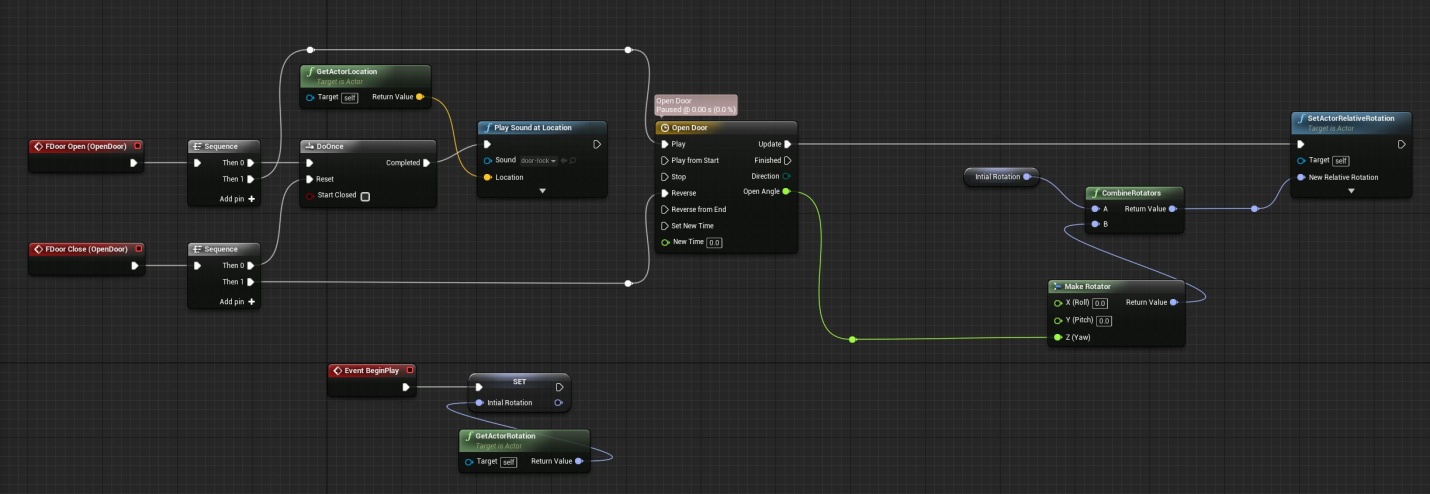
על מנת לשדר את ה – Events מתוך הקוד נקרא:

FDoorOpen.Broadcast();

פתיחת הדלת

הנה ה – Blueprint שהוא מסביר את עצמו בקלות.

הערה היחידה: שמוסיפים את - OpenAngle חשוב לשים לב שהערך השני ב – Float הוא 90 (תשעים המעלות שאנו רוצים בעצם להזיז את הדלת) שמוזן מ – 0 ל – 90 לתוך SetActorRelativeLocation.



כל Tick אנו שואלים: "האם המאסה הכוללת של התוכן ב – TriggerVolume גדולה מהמאסה הדרושה לפתיחה?".

אנחנו עושים ForLoop על TArray שלתוכו מוזן כל ה – OverlappingActors.

על על Actor מבצעים FindComponentByClass מסוג UPrimitiveComponent שמתוכו ניתן למצוא את פונקציית GetMass שמוזנת לתוך משתנה המאסה הכוללת.

במידה וישנה המאסה הרצויה אנו קוראים לדלת להיפתח ושומרים את הזמן הנוכחי בשניות GetTimeSeconds().

ואז אנו שואלים ב – Tick: "אם הזמן הנוכחי פחות הזמן האחרון שנשמר גדול מהשהיית פתיחת הדלת?" אם כן הדלת נסגרת.

**פרויקט Battle Tank**

יצירת Landscape

* Quad הינו היחידה הקטנה ביותר
* Scale קובע את גודל ה – Quad. הוא נמדד ב – CM כלומר ש – 50 שווה לחצי מטר לדוגמה (הוא במקור 1 מטר כמו קוביה ברירת מחדל במנוע)
* Section Size קובע את רשת הריבוע שיווצר, כלומר כמות ה – Quads לאורך ולרוחב (נקרא גם Component בפני עצמו)
* Number Of Components קובע את כמות הריבועים שניצור
* Number Of Components משפיע ישירות על הרזולוציה (ככל שיש יותר רכיבים כך הרזולוציה גבוה יותר מן הסתם)
* חישוב גודל השטח: Overall Resolution\*Scale=Size In Meters

לדוגמה: 2041\*0.5(50cm)=1020.5 Meters

* חשוב לזכור להוסיף LightMassImportanceVolume על מנת שהתאורה תיבנה כהלכה

צביעת Landscape

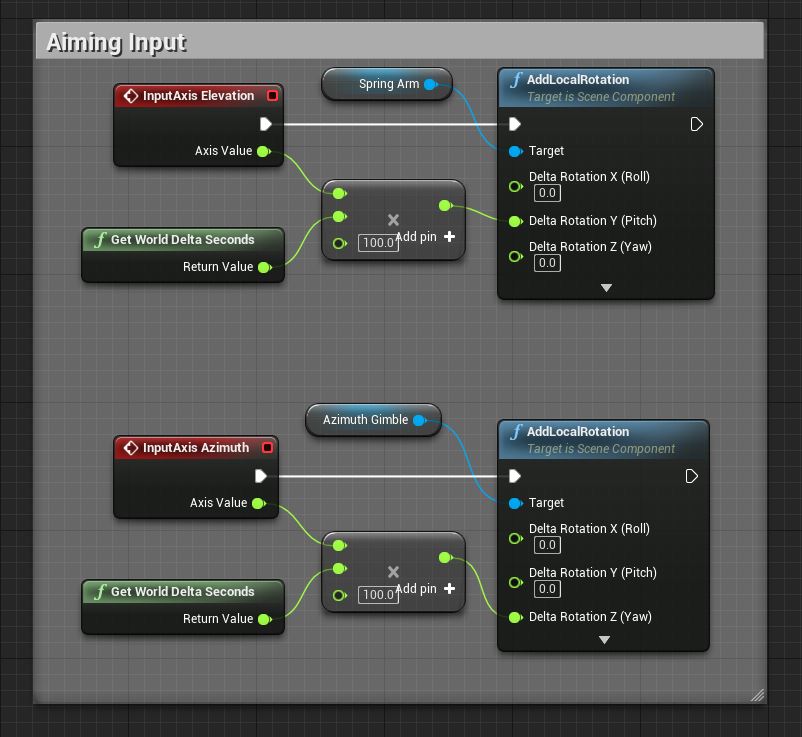
* מכינים חומר חדש
* מכניסים LandscapeLayerBlend לתוך BaseColor
* יוצרים בתוך LandscapeLayerBlend שני אלמנטים חדשים (דשא ואדמה לדוגמה)
* חובה לתת להם שם אחרת הם לא יופיע ב – Targeted Layers ב – Landscape אחרי שנוסיף לו את החומר
* נחבר אליהם Constant3Vector לצבעים או Texture Sample לטקסטורות
* במידה ומשתמשים בטקסטורות אז חובה לחבר ל – UV שלהם LandscapeLayerCoords
* נשמור את החומר
* נבחר את ה – Landscape ונשים את החומר שייצרנו בתוכו בצד ימין
* נבחר את ה – Landscape ותחת Targeted Layers יופיעו השכבות שלנו
* ליד כל שכבה ללחוץ על סימן ה - + ולבחור Weight-Blended Layer(normal) ולשמור את השכבה, לחזור על הפעולה גם לשאר השכבות

Landscape Low Poly

* מכינים Landscape חדש שהוא Low Poly
* מוסיפים לחומר הנ"ל תחת Normal את התוצאה של CrossProduct של DDX ושל DDY
* גם ה – DDX וגם ה – DDY מקבלים את ה – Input שלהם מ – Absolute World Position

הכנה מקדימה

* מכינים Pawn מ – Blueprint
* מרכיבים בתוכו את ה – Static Meshes של הטנק עם ה – Socket במקום הראוי לכל חלק
* מכניסים SprinArm כ – Child של הטנק (ולא בתור RootComponent כי היגיון מכתיב, שכאשר גוף הטנק יזוז אנו רוצים שהמצלמה תזוז איתו)
* ול – SprinArm מכניסים Child של Camera
* משנים את אורך ה – SprinArm לכמות הרצויה תחת Target Arm Length
* משנים את הרוטציה של ה – SprinArm והמצלמה עוקבת לרוטציה הזו, ככה שהרוטציה המקומית של המצלמה לא משתנה
* תחת Input בהגדרות הפרויקט מזינים את תזוזת העכבר (למעלה ולמטה ב – Axis Mapping)
* מכינים EventGraph חדש בתוך הטנק Pawn ומזינים את התזוזה לעכבר
* צריך להוסיף Scene Component שיהיה ה – Father של SpringArm עבר נעילת Gimble על ציר Azimuth



\*הנוסחה שעומדת מאחורי התזוזה פה:

Axis Value(Number between 0-1)\*Last Frame Time\*100(Sensable Sensitivity Number)

כלומר, שהתזוזה המתקבלת מהעכבר זה מספר בין 0-1, מספר קטן כשלעצמו.

הבעיה פה, שהוא מוזן בכל פריים שהמשחק מתרנדר והופך את השליטה לתלויית FrameRate.

על מנת להקטין את המספר משמעותית ולהוריד אותו מתלות ה – FrameRate יש להכפיל אותו בזמן של הפריים האחרון, מדובר על מספר קטן מאוד כ – 0.003.

כעת יש תזוזה שתלא תלויה ב – FrameRate אך התזוזה היא קטנה מאוד.

לכן נזין מכפלה נוספת של 100, שבעתיד אפשר להפוך למשתנה שהשחקן יכול לשנות ממספר מינימום למקסימום ובכך לשלוט על רגישות העכבר.

הכנה כוונת

* מכינים Widget Blueprint חדש
* בתוכו שמים תמונה לבנה ריקה בגודל הרצוי
* בפרויקט זה שמנו את העוגן 50% לרוחב המסך ו – 33% לאורכו
* סידרנו את ה – Offset של התמונה
* נכין PlayerController Blueprint חדש וב – BeginPlay שם נבנה את ה – Widget
* CreateWidget, נזין את ה – Class שיצרנו לתוכו והשחקן השולט הוא GetPlayerController
* ומיד לאחר מכן נזין את זה ל – Add To Viewport
* לא לשכוח להוסיף את ה – PlayerController החדש ל – GameMode