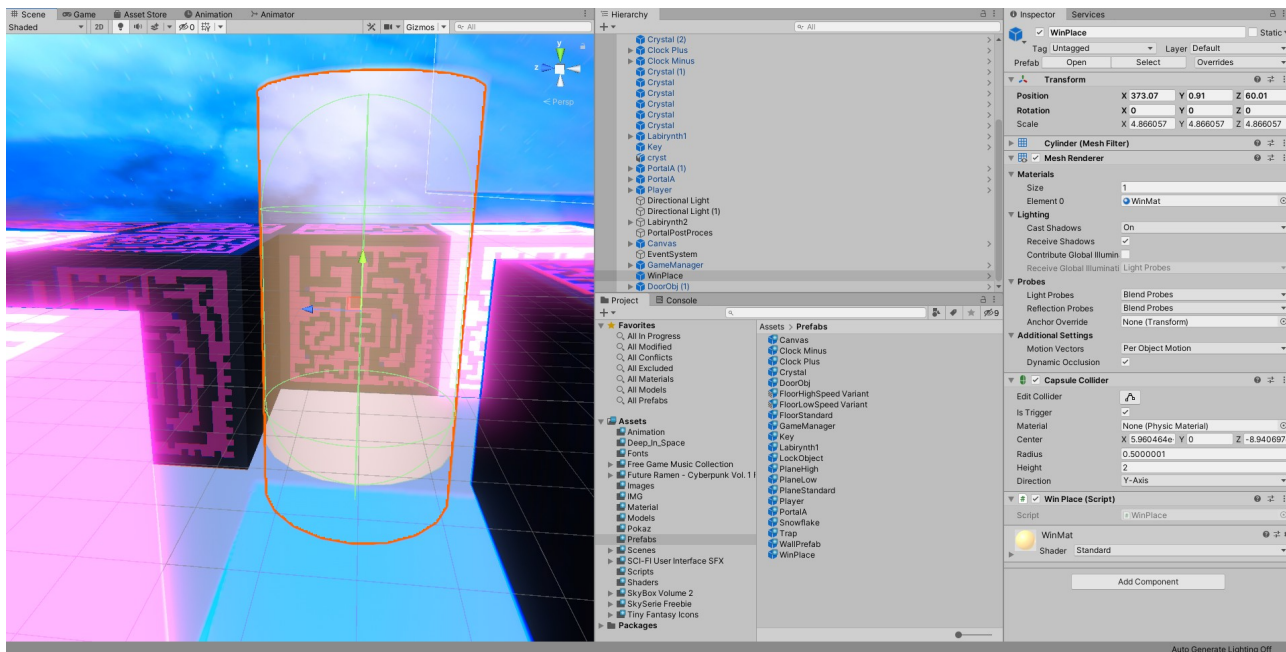


2. Zakończenie gry, kolejna rozgrywka i export

Zapewne wiele osób przygotowało sobie punkt, w którym zakończy się gra. Przed eksportem zrobimy ten punkt, by gracz jak go zobaczy, wiedział że to meta. Dodajmy sobie zwykły cylinder do sceny i powiększmy go. Dodajmy mu jakiś półprzezroczysty materiał a jego *kolider* ustawimy na **isTrigger**.



Stworzymy teraz skrypt, który z animuje ten cylinder ale też wyzwoli wygraną, po wejściu

do niego. Stwórzmy skrypt **WinPlace** i wejdźmy do niego.

Najpierw utworzymy animację. Dodamy sobie zmienną o nazwie **alfa**:

```
float alfa = 0;
```

A następnie metodę typu float o nazwie **Resizer()**. Wykorzystamy tam funkcję Sinus, która zawsze będzie zwracać w czasie ciągłym wartości z zakresu **-1** do **1**. Zastosujemy ją by powiększać nasz cylinder w osiach **x** i **z**.

```
public float Resizer()
{
    float value = Mathf.Sin(alfa);
    alfa += (1.5f * Time.deltaTime);
    return value + 2f;
}
```

Następnie w **FixedUpdate()** używamy naszej metody by stworzyć zmienną **scale** a następnie zmieniamy **localScale** naszego cylindra:

```
void FixedUpdate()
{
    float scale = Resizer();
    transform.localScale = new Vector3(scale, 4.5f, scale);
}
```

Teraz na chwilę przejdziemy do **GameManagera**. Stworzymy tam prostą metodę:

```
public void WinGame()
{
    win = true;
    endGame = true;
}
```

Będzie ona wywoływana zawsze gdy wejdziemy do **WinPlace**. Wracamy więc do niego i tworzymy metodę wywoływaną po wejściu do niego:

```
private void OnTriggerEnter(Collider other)
{
    if(other.gameObject.tag == "Player")
    {
        GameManager.gameManager.WinGame();
    }
}
```

Pozostaje nam już tylko wrócić do **GameManager**. Będziemy chcieli zrobić reload po zakończeniu gry. Musimy więc dodać możliwość przechodzenia między scenami. Dodajemy:

```
using UnityEngine.SceneManagement;
```

W metodzie Update() dodamy kawałek kodu:

```
if(endGame)
{
    if (Input.GetKeyDown(KeyCode.Y))
    {
        SceneManager.LoadScene(0);
    }

    if (Input.GetKeyDown(KeyCode.N))
    {
        Application.Quit();
    }
}
```

SceneManager.LoadScene() pozwala na otwarcie wybranej przez nas sceny. Tutaj podany jest indeks sceny (o tym za chwilę) ale można też podać jej nazwę w podwójnym cudzysłowie. **Application.Quit()** to wyłączenie programu.

Do metody **Start()** dodajemy jeszcze tylko linijkę:

```
Time.timeScale = 1f;
```

Gdyż gdy nastąpi zakończenie czas się zatrzymuje a po załadowaniu nie włącza się ponownie. Musimy więc zrobić to sami.

(Ważna UWAGA: przy eksporcie jest problem z skryptem **EditorButton**. Należy go przenieść do folderu o nazwie **Editor**. W kolejnej wersji skryptu zostanie to poprawione)

Zapisujemy. W unity klikamy przycisk **File** → **Build Settings**. Pojawia nam się okno, na górze którego jest lista **Scenes in Build**. Każda scena, której chcemy użyć w **LoadScene()** musi być na tej liście. Każda z tych list ma indeks. Scena o indeksie 0 będzie tą, która zostanie pierwsza uruchomiona.

Na dole mamy okno wyboru platformy. My zaznaczymy **PC, Mac & Linux Standalone**. W większości przypadków mamy zainstalowaną możliwość buildu tylko pod Windows, ale da się dodać możliwość eksportowania pod inne platformy z poziomu **UnityHub**. Dodajemy sceny, które chcemy by znalazły się w naszej grze a następnie klikamy **Build**. Wybieramy miejsce, do którego ma zostać eksportowana gra. Zatwierdzamy i czekamy.







Scenes In Build


☒ Scenes/TestGenerateLevels

0

[Add Open Scenes](#)

Platform

-  PC, Mac & Linux Standalone 
-  Android
-  Universal Windows Platform
- tvOS tvOS
- PS4 PS4
- iOS iOS
-  Xbox One
-  WebGL

 PC, Mac & Linux Standalone

Target Platform

Windows

Architecture

x86

Server Build

☐

Copy PDB files

☐

Create Visual Studio Solution

☐

Development Build

☐

Autoconnect Profiler

☐

Deep Profiling

☐

Script Debugging

☐

Scripts Only Build

☐

Compression Method

Default

[Learn about Unity Cloud Build](#)[Player Settings...](#)

Build

Build And Run

GameManager:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
using UnityEngine.Rendering.PostProcessing;

public class GameManager : MonoBehaviour
{
    public static GameManager gameManager;
    public int timeToEnd;
    bool gamePaused = false;
    bool endGame = false;
    bool win = false;

    public int redKey = 0;
    public int greenKey = 0;
    public int goldKey = 0;

    public int points = 0;

    AudioSource audioSource;

    public AudioClip resumeClip;
    public AudioClip pauseClip;
    public AudioClip winClip;
    public AudioClip loseClip;

    public MusicScript musicScript;

    bool lessTime = false;

    PostProcessProfile normalProfile;
    PostProcessProfile lessTimeProfile;
    PostProcessVolume volume;

    public Text timeText;
    public Text goldKeyText;
    public Text redKeyText;
    public Text greenKeyText;
    public Text crystalText;
    public Image snowFlake;

    public GameObject infoPanel;
    public Text pauseEnd;
    public Text reloadInfo;
    public Text useInfo;

    public void FreezTime(int freez)
    {
        CancelInvoke("Stopper");
        snowFlake.enabled = true; //<-----
        InvokeRepeating("Stopper", freez, 1);
    }

    public void AddPoints(int point)
    {
        points += point;
        crystalText.text = points.ToString(); //<-----
    }
}
```

```

public void AddTime(int addTime)
{
    timeToEnd += addTime;
    timeText.text = timeToEnd.ToString(); //<-----
}

public void AddKey(KeyColor color)
{
    switch (color)
    {
        case KeyColor.Red:
            redKey++;
            redKeyText.text = redKey.ToString(); //<-----
            break;
        case KeyColor.Green:
            greenKey++;
            greenKeyText.text = greenKey.ToString(); //<-----
            break;
        case KeyColor.Gold:
            goldKey++;
            goldKeyText.text = goldKey.ToString(); //<-----
            break;
    }
}

void Start()
{
    if(gameManager == null)
    {
        gameManager = this;
    }

    Time.timeScale = 1f; //<-----
    if (timeToEnd <= 0)
    {
        timeToEnd = 100;
    }

    snowFlake.enabled = false; //<-----
    timeText.text = timeToEnd.ToString(); //<-----
    infoPanel.SetActive(false); //<-----
    pauseEnd.text = "Pause"; //<-----
    reloadInfo.text = ""; //<-----
    SetUseInfo(""); //<-----
    LessTimeOff();
    audioSource = GetComponent<AudioSource>();
    InvokeRepeating("Stopper", 2, 1);
}

void Update()
{
    if(Input.GetKeyDown(KeyCode.P))
    {
        if(gamePaused)
        {
            ResumeGame();
        }
        else
        {
            PauseGame();
        }
    }

    if(endGame)
    {

```

```

        if (Input.GetKeyDown(KeyCode.Y))
        {
            SceneManager.LoadScene(0);
        }

        if (Input.GetKeyDown(KeyCode.N))
        {
            Application.Quit();
        }
    }
}

void Stopper()
{
    timeToEnd--;
    timeText.text = timeToEnd.ToString(); //<-----
    snowFlake.enabled = false; //<-----
    if (timeToEnd <= 0)
    {
        timeToEnd = 0;
        endGame = true;
    }

    if (endGame)
    {
        EndGame();
    }

    if (timeToEnd < 20 && !lessTime)
    {
        LessTimeOn();
        lessTime = true;
    }

    if (timeToEnd > 20 && lessTime)
    {
        LessTimeOff();
        lessTime = false;
    }
}

public void PauseGame()
{
    if (!endGame)
    {
        PlayClip(pauseClip);
        infoPanel.SetActive(true); //<-----
        musicScript.OnPauseGame();
        Time.timeScale = 0f;
        gamePaused = true;
    }
}

public void ResumeGame()
{
    if (!endGame)
    {
        PlayClip(resumeClip);
        infoPanel.SetActive(false); //<-----
        musicScript.OnResumeGame();
        Time.timeScale = 1f;
        gamePaused = false;
    }
}

```

```

public void EndGame()
{
    CancelInvoke("Stopper");
    infoPanel.SetActive(true); //<-----
    Time.timeScale = 0f;
    gamePaused = true;
    if (win)
    {
        PlayClip(winClip);
        pauseEnd.text = "You Win!!!"; //<-----
        reloadInfo.text = "Reload? Y/N"; //<-----
    } else
    {
        PlayClip(loseClip);
        pauseEnd.text = "You Lose!!!"; //<-----
        reloadInfo.text = "Reload? Y/N"; //<-----
    }
}

public void PlayClip(AudioClip playClip)
{
    audioSource.clip = playClip;
    audioSource.Play();
}

public void LessTimeOn()
{
    musicScript.PitchThis(1.58f);
    //volume.profile = lessTimeProfile;
}

public void LessTimeOff()
{
    musicScript.PitchThis(1f);
    //volume.profile = normalProfile;
}

public void SetUseInfo(string info)
{
    useInfo.text = info;
}

public void WinGame()
{
    win = true;
    endGame = true;
}
}

```


Lock:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Lock : MonoBehaviour
{
    public Doors[] doors;
    public KeyColor myColor;
    bool iCanOpen = false;
    bool locked = false;
    Animator key;

    AudioClip openClip;
    AudioClip lockClip;

    public Material red;
    public Material green;
    public Material gold;

    public Renderer myLock;

    private void Start()
    {
        key = GetComponent<Animator>();
        SetMyColor();
    }

    private void OnTriggerEnter(Collider other)
    {
        if(other.tag == "Player")
        {
            iCanOpen = true;
        }
    }

    private void OnTriggerExit(Collider other)
    {
        if (other.tag == "Player")
        {
            iCanOpen = false;
            GameManager.gameManager.SetUseInfo(""); //<-----
        }
    }

    private void Update()
    {
        if(iCanOpen && !locked)
        {
            GameManager.gameManager.SetUseInfo("Press E to open lock"); //<-----
        }

        if (Input.GetKeyDown(KeyCode.E) && iCanOpen && !locked)
        {
            key.SetBool("useKey", CheckTheKey());
        }
    }

    public void UseKey()
    {
        foreach(Doors door in doors)
        {

```

```

        door.OpenClose();
    }
}

public bool CheckTheKey()
{
    if(GameManager.gameManager.redKey > 0 && myColor == KeyColor.Red)
    {
        GameManager.gameManager.PlayClip(openClip);
        GameManager.gameManager.redKey--;
        GameManager.gameManager.redKeyText.text =
GameManager.gameManager.redKey.ToString(); //<-----
        locked = true;
        return true;
    }
    else if (GameManager.gameManager.greenKey > 0 && myColor == KeyColor.Green)
    {
        GameManager.gameManager.PlayClip(openClip);
        GameManager.gameManager.greenKey--;
        GameManager.gameManager.greenKeyText.text =
GameManager.gameManager.greenKey.ToString(); //<-----
        locked = true;
        return true;
    }
    else if (GameManager.gameManager.goldKey > 0 && myColor == KeyColor.Gold)
    {
        GameManager.gameManager.PlayClip(openClip);
        GameManager.gameManager.goldKey--;
        GameManager.gameManager.goldKeyText.text =
GameManager.gameManager.goldKey.ToString(); //<-----
        locked = true;
        return true;
    }
    else
    {
        GameManager.gameManager.PlayClip(lockClip);
        return false;
    }
}

void SetMyColor()
{
    switch (myColor)
    {
        case KeyColor.Red:
            GetComponent<Renderer>().material = red;
            myLock.material = red; //Tu oczywiście odpowiedni kolor
            break;
        case KeyColor.Green:
            GetComponent<Renderer>().material = green;
            myLock.material = green;
            break;
        case KeyColor.Gold:
            GetComponent<Renderer>().material = gold;
            myLock.material = gold;
            break;
    }
}
}

```

WinPlace:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class WinPlace : MonoBehaviour
{
    float alfa = 0;
    void FixedUpdate()
    {
        float scale = Resizer();
        transform.localScale = new Vector3(scale, 4.5f, scale);
    }

    public float Resizer()
    {
        float value = Mathf.Sin(alfa);
        alfa += (1.5f * Time.deltaTime);
        return value + 2f;
    }

    private void OnTriggerEnter(Collider other)
    {
        if(other.gameObject.tag == "Player")
        {
            GameManager.gameManager.WinGame();
        }
    }
}
```



Kursy przygotowawcze do egzaminu maturalnego i testu 8-klasisty z matematyki

Na koniec zajęć wyświetlamy uczestnikom stronę:

<https://www.giganciedukacji.edu.pl/>

(zapoznaj się wcześniej z ofertą na stronie tak żeby w paru zdaniach opowiedzieć swojej grupie o kursach jakie mamy w ofercie).

Tłumaczymy że oprócz nauki programowania przygotowujemy również uczestników do ważnych egzaminów z matematyki.

Zajęcia odbywają się zarówno stacjonarnie jak i online.

Nasze materiały są zgodne z podstawą programową i są przygotowywane przez egzaminatora CKE.

Na stronie uczestnicy znajdą przykładowe arkusze i rozwiązania do nich.

Zachęć do zainteresowania się ofertą kursów, wspomnij że cyklicznie organizowane są darmowe webinaria i lekcje próbne, dodatkowo warto polubić nas na facebooku, aby być na bieżąco z różnymi akcjami, które organizujemy: <https://www.facebook.com/giganciedukacji>



Prezentacja na wywiadówkę

UWAGA!!!

Jeśli prowadzisz zajęcia dla **PLACÓWKI FRANCYZOWEJ** zapytaj o szczegóły dotyczące tej prezentacji swojego przełożonego!

Wyświetlamy prezentację: <https://cutt.ly/fHWX9gF>

Opis slajdów

Na początku znajduje się slajd powitalny. Należy się przywitać i powiedzieć rodzicom po co właściwie się spotkaliśmy. Warto podkreślić, że jesteśmy nauczycielem danej grupy (bo rodzice mogą nie wiedzieć tego) i że pragniemy podsumować pracę dzieci w tym semestrze oraz przedstawić możliwości kontynuowania nauki na kolejnym semestrze. Zaznaczamy, że prezentacja nie zajmie więcej niż 15 min tak więc osoby a jeżeli ktoś się śpieszy i nie może z nami zostać to będzie możliwość nadrobienia z nagrania.

Drugim slajdem najczęściej będzie podsumowanie semestru.

Ten slajd powinien nie zająć więcej niż minuty. Możemy opisać co widzimy na prezentacji i że o wszystkim szczegółowo opowiemy w dalszej części prezentacji. Tłumaczymy też, że różnica między ilością spotkań a ilością projektów wynika z tego, że kilka projektów obejmowało więcej niż jedno spotkanie (w zależności od kursu, bo ta różnica się czasami pojawia, czasami nie).

Kolejny slajd informuje o tym jak przebiegały zajęcia. Opisujemy krótko jak wyglądały spotkania (2x45 min plus przerwa), że podczas pierwszej części mieliśmy podsumowanie materiału z poprzedniej lekcji oraz teoretyczne wprowadzenie. Następnie programujemy z jak największą ilością pracy samodzielnej dzieci. Po przerwie kontynuujemy programowanie i testujemy projekt wspólnie grając, rywalizując itp.

Dodajemy, że korzystając ze światów gier komputerowych dużo łatwiej trafiamy do dzieci z informacją edukacyjną, gdyż jest ona zakotwiczona w materiale rozrywkowym, który bardzo łatwo trafia do dzieci. Ponadto staramy się, żeby programowanie towarzyszyło dzieciom również w domu poprzez materiały powtórzeniowe i zadania.

Slajd reprezentujący ścieżkę edukacyjną

Prezentujemy jak wygląda ścieżka edukacji dla danego wieku i informujemy, że kurs kontynuowany jest w przyszłym semestrze, na który oczywiście zapraszamy.

Slajdy dot. środowisk

Slajdy dotyczące środowisk najczęściej składają się z kilku elementów. Pierwszy slajd to najczęściej grafika środowiska (logo). Drugi to slajd z ogólnymi informacjami na temat środowiska. Trzeci to przykłady projektów, jakie były wykonywane na zajęciach.

W zależności od kursu pojawiają się inne środowiska dlatego należy uważnie sprawdzać co jest na prezentacji. Informujemy rodziców co dane środowisko wprowadza i czego się dzieci mogą nauczyć (ogólnikowo, bo reszta będzie dopowiedziana dalej). Przykład dla Podstaw Tworzenia Gier Semestr 1. Poza tym co jest dodane na slajdzie dodajemy naszą wiedzę na temat środowiska informując co udało się dzieciom wykonywać, jakie elementy były podkreślane podczas zajęć i czego dzieci mogły się nauczyć przy pomocy środowiska.

Mówimy krótko, że pracowaliśmy w dwóch środowiskach – w Scratchu poznawaliśmy podstawy języków blokowych i podstawowe definicje programistyczne, a w Minecrafcie uczyliśmy się wykorzystywać wiedzę w praktycznym kontekście (edycja świata już istniejącej gry) oraz poznawaliśmy bardziej zaawansowane pojęcia z pogranicza programowania, informatyki i matematyki.

Pokazując projekty (w formie filmików) należy skupić się tylko na najważniejszych informacjach. Nie opisujemy filmików zbyt dokładnie. Najważniejsze to pokazać filmikiem jak wyglądało programowanie i trochę o tym opowiedzieć oraz jak wyglądał sam efekt pracy. Opisujemy krótko na czym gra polega ale bez wdawania się w szczegóły.

Slajd z wartościami edukacyjnymi

Slajd przedstawia umiejętności jakie dziecko pozyskuje na naszych zajęciach. Bardzo ważne aby krótko o nich opowiedzieć. Przykładowo jak w Waszej pracy jako programista pomaga logiczne myślenie czy wykorzystywanie i przetwarzanie informacji. Przykłady umiejętności (nie wszystkie) i co można opowiedzieć poniżej:

Logiczne myślenie – uczestnicy muszą szukać powiązań między skryptami a efektem, a w przypadku błędu uczymy się go lokalizować i rozwiązać

Obsługa komputera i programów – szczególnie ważna wśród młodszych dzieci.

Przykładowo jest edytor graficzny duszków dający podstawy pracy z programami graficznymi a u starszych np. modelowanie obiektów graficznych w formie 3D

Myślenie kreatywne – uczniowie często mają możliwość dodania własnych elementów do gry, programu lub edytować różne opcje według własnego uznania.

Umiejętność samodzielnej pracy – uczestnicy często muszą wykazywać się pracą indywidualną ale zdarzają się również momenty pracy grupowej

Zdolności matematyczne i umiejętności analityczne – przeliczanie zmiennych, koordynatów itp.

Znajomość języków blokowych – czyli w zasadzie poznanie podstaw programowania

Wykorzystywanie i przetwarzanie informacji – na podstawie przekazywanej informacji od nauczyciela uczniowie skłonieni są do stworzenia jakiegoś skryptu i zaprezentowania jego działania

Dlaczego warto programować

Poza tym co znajduje się na slajdzie warto opowiedzieć o pracy programisty jeżeli się nią zajmujecie (np. jakieś informacje na temat jak to wygląda na co dzień – oczywiście w samych superlatywach). Można podkreślić, że dziecko czeka bezpieczna przyszłość jako programista, a nawet jeżeli dziecko nie będzie programistą to nauka programowania pomoże dzieciom w rozumieniu i uczeniu się przedmiotów ścisłych (dzięki rozwijaniu różnych kluczowych kompetencji szkolnych). Warto dodać, że programowanie to nie tylko praca programisty ale przydaje się również w innych zawodach – analitycy, marketingowcy, graficy komputerowi itp.

Slajd o kontynuacji ścieżki

Informujemy o kontynuacji kursu na następnym semestrze. Jak to zrobić i co czeka na nas w następnym semestrze pokażemy za chwilę.

Slajd z następnym semestrze

Z jak największym zaangażowaniem prezentujemy co będziemy robić w następnym semestrze. Poza tym co jest na slajdzie posilkujemy się stroną internetową i opowiadamy co dzieci będą robić w następnym semestrze. Pokazujemy również przykłady projektów jakie są tam zamieszczone w formie filmików.

Slajd ze schematem kontynuacji

Aby kontynuować naukę wystarczy wejść na maila, który wysyłamy 1-2 tyg przed zakończeniem zajęć. Jest to mail ze wszystkimi informacjami na temat następnego kursu. Znajduje się tam przycisk o chęci kontynuacji, który należy kliknąć. Wtedy też zostanie wygenerowana umowa, którą należy podpisać, wpłacić zaliczkę w wysokości 100 zł i możemy się cieszyć kolejnym semestrem. Ponadto przygotowaliśmy pewną nagrodę dla każdego, kto zrobi to w ciągu 2 tyg od momentu otrzymania maila (następny slajd)

Slajd z promocją na kursy krótkie

Poza tym co jest na slajdzie dodajemy, że na grafice są tylko niektóre kursy ale promocja dotyczy wszystkich krótkich kursów wakacyjnych. Krótkie kursy mają po 5 spotkań, które odbywają się w ciągu jednego tygodnia i rozszerzają wiedzę z konkretnego zakresu.

Slajd ostatni

Informujemy, że w przypadku pytań pozostajemy do dyspozycji. Jeżeli już teraz wiedzą, że np. nie mogą się zapisać bo jakiś basen czy coś to niech koniecznie skontaktują się z nami. Postaramy się dopasować dzień i godzinę do indywidualnych potrzeb rodzica (i mówimy o tym tak, żeby rodzice czuli, że poruszymy niebo i ziemię żeby dziecko miało możliwość uczestnictwa w zajęciach tak, żeby rodzice nie poczuli się osamotnieni z problemem). Ponadto zapraszamy na Discorda, bo w wakacje będzie się trochę tam działo (m.in. zadanka).

Kilka porad:

Mówiąc o Minecrafcie zawsze używamy pełnej nazwy MINECRAFT EDUCATION EDITION.

Warto podkreślić, że pokazujemy zakulisowy charakter rozrywki (gier, programów, aplikacji) od strony twórcy, a nie odbiorcy. Ponadto pokazujemy jak wykorzystać czas, który zwykle dzieci wykorzystują na granie w pożyteczny, edukacyjny sposób.

Mówiąc o Minecrafcie można dodać, że jesteśmy oficjalnym partnerem Microsoftu.