Lekcja 18. Tajniki budowniczego Robloxa - Specjalne skrzynki

Cel lekcji

Celem lekcji będzie wprowadzenie dodatkowych skrzynek z atrakcjami do rozgrywki.

Agenda

1. Pytania początkowe.
2. Programujemy spadające skrzynki.
3. Testy.
4. Pytania końcowe i podsumowanie lekcji.
5. Wywiadówka

1. Pytania początkowe

1. Co stworzyliśmy na ostatnich zajęciach?

2. Czy ktoś z was słyszał o spadających skrzynkach w grach?

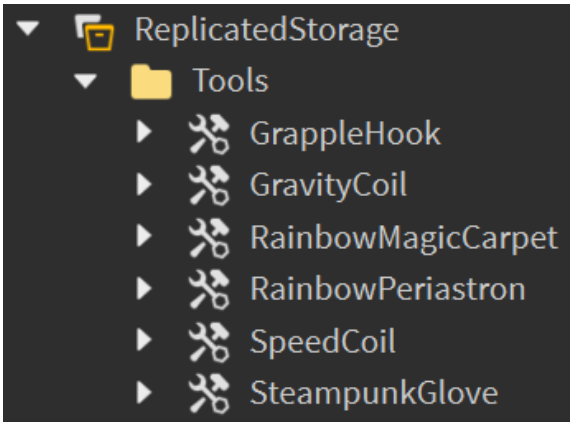
3. Co można zdobyć z takich skrzynek?

2. Programujemy spadające skrzynki

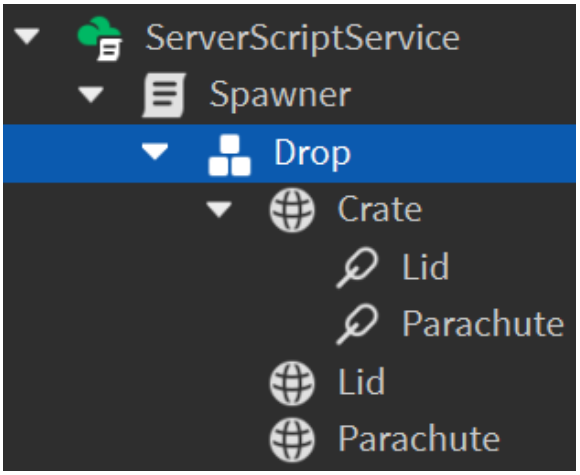
Uruchamiamy specjalnie przygotowany projekt startowy.

Projekt startowy (na dysku)

Narzędzia, które znajdują się w folderze **Tools**, posłużą za dodatkową atrakcję do rozgrywki, taki system można dodać do wielu gier, w formie zabawy.



Model **Drop**, który znajduje się w skrypcie **Spawner**, to specjalna skrzynia ze spadochronem, który będzie usuwany po dotarciu skrzyni na ląd.



Dodatkowy opis do LinearVelocity oraz Angular Velocity:

LinearVelocity i AngularVelocity są częścią API gry Roblox i służą do kontroli ruchu obiektów w grze.

LinearVelocity jest używane do nadawania liniowego ruchu obiektowi. Oto kilka z jego właściwości:

- MaxForce: Określa maksymalną siłę, z jaką obiekt może być przemieszczany. W przypadku skryptu ustawione jest na math.huge, co oznacza, że nie ma praktycznego limitu.

- RelativeTo: Określa, do czego jest względna prędkość. Może to być Świat (World), Lokalny (Local), czy Attachment0.

- VelocityConstraintMode: Definiuje, jak obiekt jest przemieszczany. Może to być wektor (Vector), który przemieszcza obiekt wzdłuż linii, lub skierowany (Directed), który przemieszcza obiekt w kierunku innego obiektu.

- VectorVelocity: Określa prędkość i kierunek przemieszczania obiektu.

AngularVelocity jest używane do nadawania obrotu obiektowi. Jego właściwości to:

- MaxTorque: Określa maksymalny moment obrotowy, z jakim obiekt może być obracany. Podobnie jak MaxForce, jest ustawione na math.huge.

- RelativeTo: Podobnie jak w LinearVelocity, określa, do czego jest względny ruch obrotowy. Może to być Świat (World), Lokalny (Local), czy Attachment0.

- AngularVelocity: Określa prędkość i kierunek obrotu obiektu.

W skrypcie te klasy są używane do symulowania upadku skrzynki z narzędziami. **LinearVelocity** jest używane do nadawania skrzynce prędkości upadku, podczas gdy **Angular Velocity** jest używane do nadawania skrzynce obrotu, aby upadek wyglądał bardziej realistycznie.

Opis do programu:

Ten program generuje losowe "dropy" w postaci skrzynek na mapie gry. Każda skrzynka zawiera jeden losowy przedmiot z zestawu narzędzi przechowywanych w zasobach zreplikowanych gry.

Oto szczegółowy opis:

1. Skrypt tworzy nową instancję klasy `Random` i używa jej do generowania liczb losowych.

2. Ustalane są minimalne i maksymalne wartości czasu dla dropów skrzynek oraz minimalna i maksymalna prędkość upadku skrzynek.

3. Skrypt pobiera wszystkie narzędzia z magazynu zreplikowanych gry.

4. W nieskończonej pętli skrypt wykonuje następujące czynności:
   - Czeka losowy czas (między wartościami **minTimePerDrop** i **maxTimePerDrop**).
   - Klonuje obiekt **Drop** (reprezentujący skrzynkę) i umieszcza go na losowej pozycji na mapie, obrócony o losowy kąt.
   - Tworzy nowe instancje klas "LinearVelocity" i "AngularVelocity" i używa ich do przemieszczania i obracania skrzynki. Prędkość przemieszczania i obracania jest losowa, ale jest ograniczona do wartości **minFallVelocity** i **maxFallVelocity**.
   - Po zetknięciu się skrzynki z graczem, skrzynka jest niszczona, a gracz otrzymuje losowe narzędzie z magazynu gry. Narzędzie jest dodawane do plecaka gracza.
   - Po zakończeniu upadku, skrzynka jest niszczona, a elementy, które były używane do jej przemieszczania i obracania, są usuwane.
   - Spadochron skrzynki jest niszczony, a usługa **Debris** gry jest informowana o tym, aby usunęła spadochron po upływie 5 sekund.

Podsumowując, ten program tworzy system upadków skrzynek, gdzie losowe skrzynki są generowane na mapie w określonych odstępach czasu i gracze mogą je zbierać, otrzymując losowe przedmioty.

#tools - oznacza ilość elementów (narzędzi) # długość

Przechodzimy do programowania skryptu **Spawner**.

(Program jest w fazie testowej, później zostanie maksymalnie uproszczony, jeżeli jest słabsza grupka część programu wysyłamy i tłumaczymy).

Link do Pastebin: https://pastebin.com/4iNxHfD7

Program bez komentarzy:

```
local rnd = Random.new()

local minTimePerDrop = 2
local maxTimePerDrop = 6

local minFallVelocity = -5
local maxFallVelocity = -15

local tools = game.ReplicatedStorage.Tools:GetChildren()

while true do

   task.wait(rnd:NextNumber(minTimePerDrop, maxTimePerDrop))

   local drop = script:WaitForChild("Drop"):Clone()

   local randomPos = Vector3.new(
      rnd:NextNumber(-512, 512),
      200,
      rnd:NextNumber(-512, 512))
   local newCF = CFrame.new(randomPos) * CFrame.Angles(0, rnd:NextNumber(0, 2 * math.pi), 0)
   drop:PivotTo(newCF)

   local atch0 = Instance.new("Attachment")
   atch0.Name = "Attachment0"
   atch0.Parent = drop.Crate

   local lv = Instance.new("LinearVelocity")
   lv.MaxForce = math.huge
   lv.RelativeTo = Enum.ActuatorRelativeTo.World
   lv.VelocityConstraintMode = Enum.VelocityConstraintMode.Vector
   lv.VectorVelocity = Vector3.new(
      rnd:NextNumber(-5, 5),
      rnd:NextNumber(maxFallVelocity, minFallVelocity),
      rnd:NextNumber(-5, 5))
   lv.Attachment0 = atch0
   lv.Parent = drop.Crate

   local av = Instance.new("AngularVelocity")
   av.MaxTorque = math.huge
   av.RelativeTo = Enum.ActuatorRelativeTo.Attachment0
   av.AngularVelocity = Vector3.new(0, rnd:NextNumber(-1, 1), 0)
   av.Attachment0 = atch0
   av.Parent = drop.Crate


   drop.Crate.Touched:Connect(function(hit)
```

```
        local plr = game.Players:GetPlayerFromCharacter(hit.Parent)

        if plr and hit.Parent.Humanoid.Health > 0 then

            drop:Destroy()

            local random = math.random(1, #tools)

            local newTool = tools[random]:Clone()

            newTool.Parent = plr.Backpack
        end
    end)

    drop.Parent = workspace

    task.spawn(function()
        repeat
            task.wait(1)
        until not drop or drop.Parent ~= workspace or drop.Crate.AssemblyLinearVelocity.Y >
minFallVelocity

        if drop and drop.Parent == workspace then
            lv:Destroy()
            av:Destroy()
            atch0:Destroy()

            drop.Crate.Parachute:Destroy()
            drop.Parachute.CanCollide = false

            game:GetService("Debris"):AddItem(drop.Parachute, 5)
        end
    end)
end
```

Program z komentarzami:

```
-- Tworzy nową instancję klasy Random
local rnd = Random.new()

-- Określa minimalny i maksymalny czas pomiędzy dropami
local minTimePerDrop = 2
local maxTimePerDrop = 6

-- Określa minimalną i maksymalną prędkość upadku skrzynek
local minFallVelocity = -5
local maxFallVelocity = -15

-- Pobiera wszystkie narzędzia z magazynu zreplikowanych gry
```

```lua
local tools = game.ReplicatedStorage.Tools:GetChildren()

-- Nieskończona pętla generująca dropy
while true do
    -- Czeka losowy czas między dropami
    task.wait(rnd:NextNumber(minTimePerDrop, maxTimePerDrop))

    -- Klonuje obiekt "Drop"
    local drop = script:WaitForChild("Drop"):Clone()

    -- Ustala losową pozycję dla dropu
    local randomPos = Vector3.new(rnd:NextNumber(-512, 512), 200, rnd:NextNumber(-512, 512))
    -- Tworzy nowy układ współrzędnych (CFrame) dla dropu
    local newCF = CFrame.new(randomPos) * CFrame.Angles(0, rnd:NextNumber(0, 2*math.pi), 0)
    -- Przemieszcza drop do nowego układu współrzędnych
    drop:PivotTo(newCF)

    -- Tworzy nowe załączniki (attachments) dla dropu
    local atch0 = Instance.new("Attachment")
    atch0.Name = "Attachment0"
    atch0.Parent = drop.Crate

    -- Ustala prędkość liniową dropu
    local lv = Instance.new("LinearVelocity")
    lv.MaxForce = math.huge
    lv.RelativeTo = Enum.ActuatorRelativeTo.World
    lv.VelocityConstraintMode = Enum.VelocityConstraintMode.Vector
    lv.VectorVelocity = Vector3.new(rnd:NextNumber(-5, 5), rnd:NextNumber(maxFallVelocity, minFallVelocity),
rnd:NextNumber(-5, 5))
    lv.Attachment0 = atch0
    lv.Parent = drop.Crate

    -- Ustala prędkość kątową dropu
    local av = Instance.new("AngularVelocity")
    av.MaxTorque = math.huge
    av.RelativeTo = Enum.ActuatorRelativeTo.Attachment0
    av.AngularVelocity = Vector3.new(0, rnd:NextNumber(-1, 1), 0)
    av.Attachment0 = atch0
    av.Parent = drop.Crate

    -- Po zetknięciu się dropu z graczem, drop jest niszczony, a gracz otrzymuje losowe narzędzie
    drop.Crate.Touched:Connect(function(hit)
        local plr = game.Players:GetPlayerFromCharacter(hit.Parent)

        if plr and hit.Parent.Humanoid.Health > 0 then
            drop:Destroy()
            -- # oznacza długość (ilość elementów)
            local random = math.random(1, #tools)

            local newTool = tools[random]:Clone()

            newTool.Parent = plr.Backpack
```

```
        end
    end)

    -- Umieszcza drop w przestrzeni roboczej (workspace)
    drop.Parent = workspace

    -- Po zakończeniu upadku, niszczy drop i wszystko, co z nim związane
    task.spawn(function()
        repeat
            task.wait(1)
        until not drop or drop.Parent ~= workspace or drop.Crate.AssemblyLinearVelocity.Y > minFallVelocity

        if drop and drop.Parent == workspace then
            lv:Destroy()
            av:Destroy()
            atch0:Destroy()

            drop.Crate.Parachute:Destroy()
            drop.Parachute.CanCollide = false

            game:GetService("Debris"):AddItem(drop.Parachute, 5)
        end
    end)
end
```

2.1* Zadanie dodatkowe

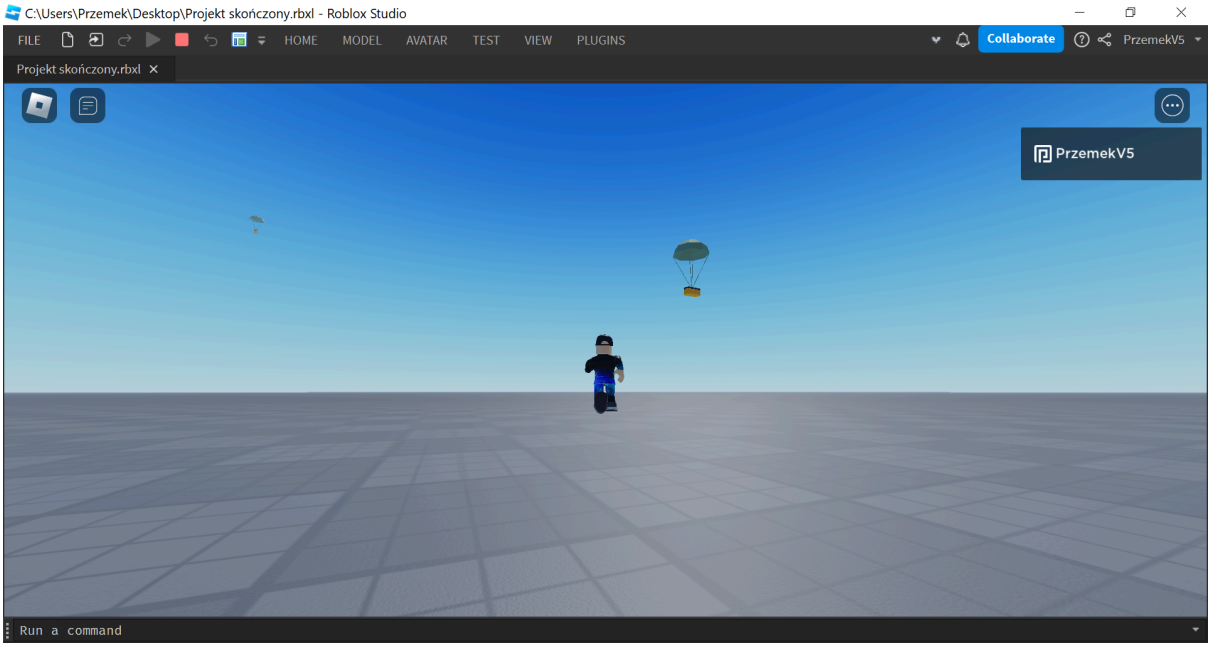
Uczestnicy samodzielnie próbują dodać dodatkowe narzędzia z **Toolboxa**, należy je umieścić w folderze **Tools** w **ReplicatedStorage.**

2.2* Zadanie dodatkowe

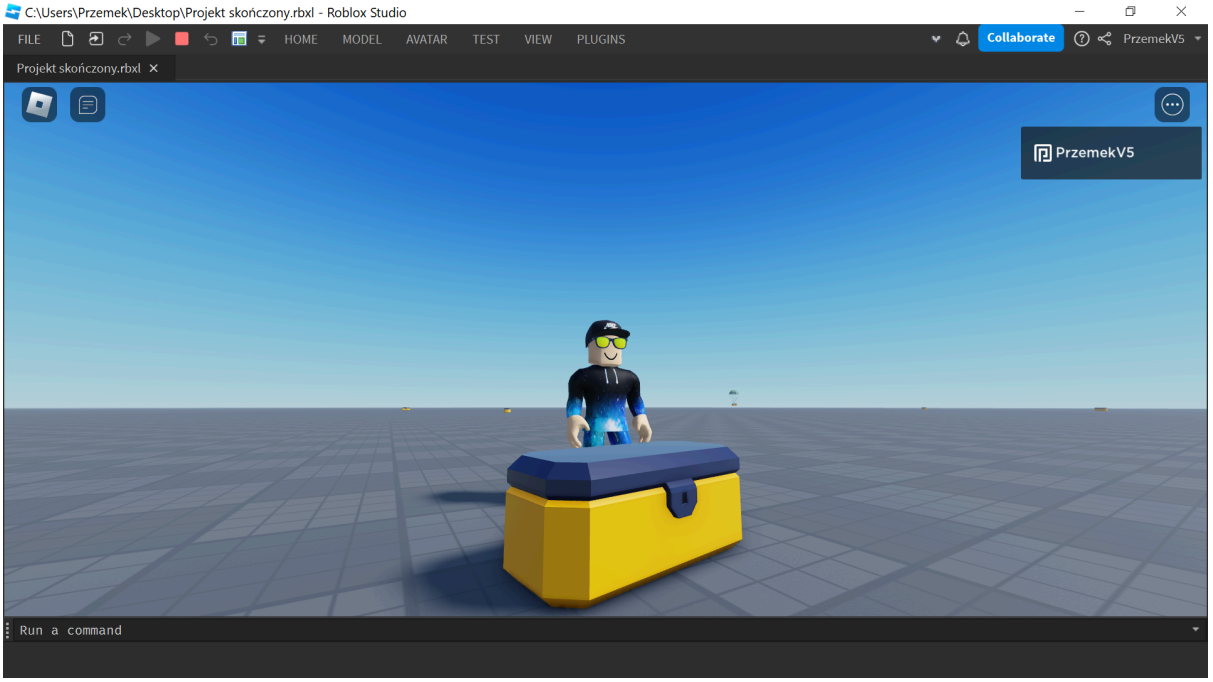
Wspólne testy, publikujemy grę i wchodzimy wszyscy na nią (wysyłamy linka do gry na czacie).

3. Testy

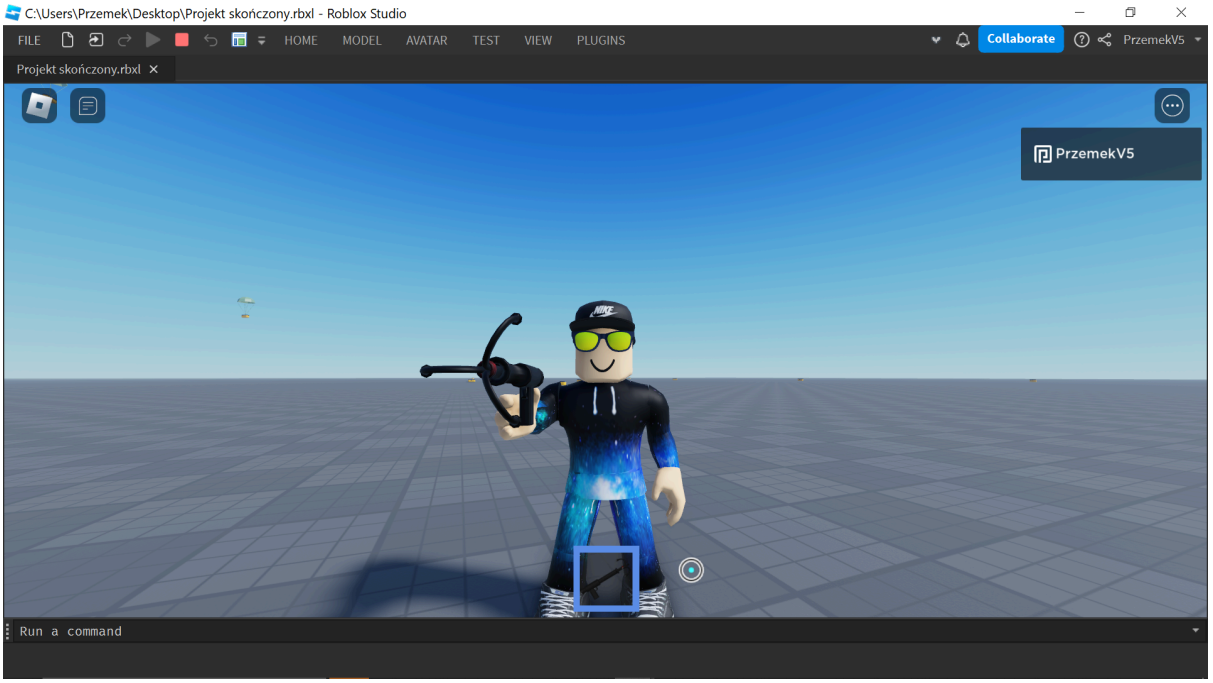
Po uruchomieniu gry skrzynki powinny zacząć spadać z nieba.



Po dotarciu na ziemię, spadochron powinien się odczepić i opaść.



Po dotknięciu skrzyni, powinniśmy dostać nowe narzędzie.



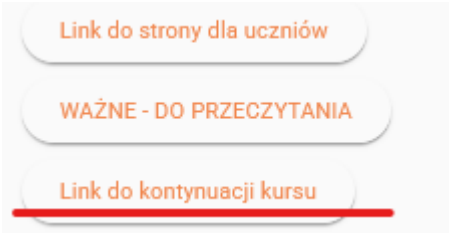
4. Pytania końcowe i podsumowanie lekcji

1. Czego dzisiaj się nauczyliśmy?

2. Co można dodać do wygrania w skrzyniach?

3. Jakie macie pomysły na dodatkowe rzeczy w grze, które będą dodawać atrakcje do gry?

Kontynuacja kursu
W przypadku jeśli ta lekcja jest ostatnią dla danej grupy należy pokrótce opowiedzieć o kursie, który jest kontynuacją dla obecnego. Link z opisem znajduje się w liście obecności:



Należy również opowiedzieć o procesie kontynuacji kursu. Opis znajduję się ogłoszeniu: https://discord.com/channels/695225372265939015/973553136851623936/1374405710158364762