# Bakgrund och fakta

## Block- och flödesprogrammering

I ett programmeringsspråk där översättning till andra språk prioriteras finns det ett antal olika tillvägagångssätt, alla med olika för- och nackdelar.

Block programmering bland det vanligaste av de olika typerna av översättningsbara programmeringsspråk. Dessa språk är ofta till för utbildningssyfte och fungerar på sätt att de består utav block, liknande dom i Lego. Blocken kan ha olika funktioner som till exempel värden, uttryck eller olika satser. Man bygger ett program genom att koppla ihop dessa block i en specialbyggd programmeringsmiljö. De kändaste och mest använda programmeringsspråket (Scratch Foundation, 2021b) implementerat på detta vis är Scratch (Scratch Foundation, 2021a), originellt utvecklat av MIT senare Scratch Foundation. Detta språk utvecklades i utbildningssyfte med översättning och lättanvändning i åtanke och dess målgrupp är framförallt grundskolebarn.

Flödes- eller nodbaserade programmeringsspråk är egentligt på många sätt lika de block baserade i att dem använder sig utav visuella block. I ett flödesbaseratpråk är dock dessa block inte sekventiella utan ihopkopplade med sladdar som representerar ett värdes flöde genom programmet. Spelmotorer som Unreal (Epic Games, 2021), Unity (Unity Technologies, 2021) och Godot (Godot Engine contributors, 2021) samt 3d modelleringsprogram som Blender (Blender Foundation, 2021) använder sig av nodbaserad programmering för ett enklare alternativ till traditionell textbaserad programmering. Detta alternativ är likt blockprogrammering i det att etiketterna och texterna på noderna är översättningsbara.

Problemet som uppkommer för både flödes- och blockbaserad programmering är det att dom kräver en integrerad utvecklingsmiljö så som en hemsida för Scratch eller respektive programs skrivbords app för Unreal, Unity, Godot och Blender. Att utveckla en integrerad utvecklingsmiljö utöver ett programmeringsspråk och dess beståndsdelar skulle utgöra ytterligare problem och är egentligen utanför projektets omfattning.

## Text

Det andra alternativet till visuella programmeringsspråk så som de block- och flödesbaserade språken tidigare nämnda är ett mer traditionellt textbaserat språk. Det finns många textbaserat programmeringsspråk och dom flesta är textbaserade men väldigt få uppfyller kravet att vara översättningsbart.

Ett tidigt exempel på ett översättningsbart programmeringsspråk är det av ALGOL 68. Detta språk standardiserades och dess standard publicerades i flera olika naturliga språk. Denna standard översattes till ryska, tyska, franska, bulgariska och senare japanska samt en för dess använding i blindskrift. Detta ledde till att standarden antogs och accepterades av både UNESCOs organisation IFIP samt Sovjets och senare Rysslands standardorganisation.

Citrine är ett programmeringsspråk där lokalisation är en av kärnfunktionerna vilket har lett till dess översättning till 111 olika naturliga språk. Språket lokaliserar nyckelord, nummer och skiljetecken. För att konvertera mellan olika språk kan användaren själv översätta programmet men inget inbyggt verktyg i programmet verkar göra detta åt användaren. Detta då alla olika naturliga språk som Citrine stödjer publiceras som separata program utan vetskapen om hur man skulle översätta ett program från ett till ett annat naturligt språk.

Scheme är ännu ett standardiserat programmeringsspråk med möjlighet till internationalisation. Detta är dock inte en kärnfunktion i språket utan har istället utvecklats av användare som har använt språkets flexibilitet för att skapa ett bibliotek där olika översättningar finns. Eftersom olika språk kan laddas dynamisk går Scheme program att vara flerspråkiga. Detta leder dock till den nackdelen att språket ej enkelt kan översättas då flera olika språk skulle kunna existera i samma program samt det faktum att språket inte är byggt med översättning, lokalisation eller internationalisation som en kärnfunktion.

## Lexikalanalys

## Syntaxanalys

## Körning

## Källförtäckning

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The Scratch Programming Language and Environment. ACM Transactions on Computing Education, 10(4), 1–15. doi:10.1145/1868358.1868363

Kuhail, M. A., Farooq, S., Hammad, R., & Bahja, M. (2021). Characterizing Visual Programming Approaches for End-User Developers: A Systematic Review. IEEE Access, 9, 14181–14202. doi:10.1109/access.2021.3051043

Jost, B., Ketterl, M., Budde, R., & Leimbach, T. (2014). Graphical Programming Environments for Educational Robots: Open Roberta - Yet Another One? 2014 IEEE International Symposium on Multimedia. doi:10.1109/ism.2014.24

O’Hara, K., & Kay, J. (2003). Investigating Open Source Software and Educational Robotics. J. Comput. Sci. Coll., 18(3), 8–16.

Papert, S. (1980). Mindstorms: Children, Computers, and Powerful Ideas. Basic Books, Inc..

Blender Foundation. (2021). *Introduction — Blender Manual*. <https://docs.blender.org/manual/en/latest/render/shader_nodes/introduction.html>

Epic Games. (2021). *Blueprint Visual Scripting*. <https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints>

Godot Engine contributors. (2021). *VisualScript*. <https://docs.godotengine.org/en/stable/getting_started/scripting/visual_script/index.html>

Scratch Foundation. (2021a). *Scratch - About*. <https://scratch.mit.edu/about>

Scratch Foundation. (2021b). *Scratch - Statistics*. <https://scratch.mit.edu/statistics>

Unity Technologies. (2021). *Bolt Visual Scripting Unity*. <https://unity.com/products/unity-visual-scripting>