Specifikace softwarového díla

&
Časový plán implementace

pro

Boxer-HTML

Boxer-HTML je programovací systém věrný historickému programovacímu systému Boxer, původně vytvořený v jazyce Lisp. Boxer-HTML vznikl jako snaha analyzovat a implementovat unikátní vlastnosti tohoto systému, popsány dále ve specifikaci.

1.0.1

Jan Markvart

17.9.2024

Obsah

Tabulka revizí

Jméno	Datum	Důvod změny	Verze
Jan Markvart	5.8.2024	Počáteční text	0.1.0
Jan Markvart	25.8.2024	Doplněny požadavky, rozhraní	0.2.0
Jan Markvart	5.9.2024	Doplněn popis funkcionality	0.3.0
Jan Markvart	13.9.2024	Revize specifikace	1.0.0
Jan Markvart	17.9.2024	Oprava sekce 1.3 - Odkazy	1.0.1

1. Základní informace

1.1 Popis a zaměření softwarového díla

Boxer je historický programovací systém původně vytvořený v jazyce Lisp. Od většiny dnešních programovacích systémů se rozlišuje dvěma vlastnostmi:

1) Spatial metaphor – celý Boxer program si lze představit jako 2D plochu obsahující vlastnosti mezi objekty, nazývané boxy. Box je sekce tohoto prostoru obsahující text, další boxy, nebo libovolnou kombinaci obojího. Tímto vzniká na ploše hierarchie programu. Uživatel s požadovanou sekcí interaguje výběrem pomocí kurzoru myši.
2) Naive realism – naivní realismus pracuje s iluzí, že uživatel vidí jádro programu a jeho stav po celou dobu jeho tvorby a exekuce (tento koncept je typický pro textové editory, ale vzácně se vyskytuje v programovacích jazycích). Všechen text programu, ať vytvořený uživatelem nebo systémem, je tedy přístupný a kdykoli editovatelný. Uživatel tak může vidět stav programu a upravit ho pouhou změnou textu v daném boxu.

Programovací systém Boxer-HTML je jednoduchý grafický programovací systém, vytvořený nad technologiemi typickými pro webové stránky. Systém je zaměřen na lidi se zájmem o historii unikátních programovacích systémů jako byl Boxer, jehož koncepty co nejvěrněji napodobuje v modernějším a přístupnějším webovém rozhraní.

Aktuálně existuje alternativní re-implementace v jazyku Lisp, ve kterém byl vytvořen i původní Boxer. naše řešení se rozlišuje převážně runtime prostředím (webové namísto samostatného grafického okna) a s ním spojenou snahou o rozšiřitelnost v rámci jiných webů.

1.2 Použité technologie

Projekt používá standardní JavaScript, HTML a kaskádové styly CSS.

1.3 Odkazy (Reference)

PDF obsahující popis původního programovacího systému Boxer:

https://dl.acm.org/doi/10.1145/6592.6595, A. A diSessa, H. Abelson, 1986

Odkaz na aktuální iteraci systému Boxer v jazyku Lisp:

https://boxer-project.github.io, Boxer Sunrise team

2. Stručný popis softwarového díla

2.1 Důvod vzniku softwarového díla a jeho základní části a cíle řešení

Projekt vznikl jako snaha re-implementovat část starého programovacího systému Boxer v moderním webovém prostředí. Aktuálně se zaměřuje na funkcionalitu spojenou s kreslícími operacemi nad turtle graphics, v Boxer-HTML využívající html <canvas> tag, práci s číselnými proměnnými u kreslících operací, a interakce mezi boxy – volání boxu z jiného, a předávání proměnných a argumentů.

2.2 Hlavní funkce

Uživatel bude v rámci používání systému definovat boxy s jednoduchou funkcionalitou, a pomocí stavění na dříve definovaných boxech zvyšovat komplexitu boxů, a tím i celkového programu.

Programovací systém bude tedy umožňovat:

- Vytváření boxů (základní jednotky programu)
- interakce mezi boxy (vnoření, volání, reference mezi boxy)
- Využívání kreslících operací k vykreslování na canvas

2.3 Motivační příklad užití

Při správně nadefinovaných pravidlech v boxech může uživatel vykreslovat libovolné fraktály, či jiné geometrické obrazce.

2.4 Prostředí aplikace

Programovací systém pracuje v běžném webovém prohlížeči. Ke spuštění je třeba pouze otevřít příslušný html soubor. Interpreter je vytvořen v jazyce Javascript bez využití jakýchkoliv knihoven, není tedy třeba žádné další specifické prostředí.

2.5 Omezení díla

Programovací systém bude pracovat pouze s číselnými proměnnými (rozšíření na textové proměnné je plánováno v navazující bakalářské práci). Programovací systém se v rámci ročníkového projektu omezuje na funkcionalitu popsanou v odkazu č.1 pod figure 2 (základní chování boxů, kreslící operace).

3. Vnější rozhraní

3.1 Uživatelské rozhraní, vstupy a výstupy

Uživatel program upravuje přímo pomocí content-editable vlastnosti html tagů, s pomocí klávesových zkratek může dynamicky vytvářit nové boxy.

Program nemá typicky známe vstupy, namísto toho se „vstupní“ proměnné zapisují přímo do relevantních boxů, a jejich hodnota se použije při spuštění daného boxu.

Výstup je obdobně uzavřen v rámci programu, v aktuální verzi jako vykreslené obrazce v html <canvas> tagu.

4. Detailní popis funkcionality

4.1 Tvorba a editace boxů

Box existuje jako základní stavební prvek programu, obdobně jako například funkce ve funkcionálních jazycích. K vytváření boxů uživatel využívá klávesové zkratky.

Editace názvu a kódu jednotlivých boxů se provádí kliknutím na příslušnou oblast boxu, čímž se spustí content-editable vlastnost HTML a uživatel může boxy upravovat.

Boxy se dělí na spustitelné doit-boxy a nespustitelné data-boxy.

4.2 Definice „primitivních“ operací

Jako primitivní operaci definujeme operaci, která není boxem. V aktuální verzi se jedná o kreslící operace forward, skip, rotate, operaci repeat a volání jiných boxů. Primitivní operace jsou popsány níže:

1) **forward**
   - argument 1: číselná hodnota n
   - primitivní operace **forward** posune kreslící hlavu na canvasu o n pixelů, zanechávající za sebou rovnou čáru.
2) **skip**
   - argument 1: číselná hodnota n
   - Primitivní operace **skip** posune kreslící hlavu na canvasu o n pixelů, bez vykreslení čáry.
3) **rotate**
   - argument 1: číselná hodnota n
   - Primitivní operace **rotate** rotuje kreslící hlavu o n stupňů ve směru hodinových ručiček.
4) **repeat**
   - argument 1: číselná hodnota n
   - argument 2: doit-box nebo název, kterým odkazuje na doit-box
   - primitivní operace **repeat** zopakuje n-krát spuštění příslušného doit-boxu.

V případě výjimky se příslušná operace neprovede a program postupuje na další operaci.

4.3 Spuštění doit-boxu

Uživatel doit-box spustí jednoduchým klikem na jeho vnější oblast, čímž se postupně začnou vykonávat všechny operace v něm obsažené, v pořadí tak jak jsou definovány.

4.4 Proměnné

Všechny proměnné jsou uloženy v zásobníku. Nové proměnné jsou tedy přidávány na vrchol, a při vyhledávání se berou v pořadí nejnovější až nejstarší. To mimo jiné znamená, že během běhu může vzniknout několik proměnných se stejným názvem, v takovém případě se bere vždy ta nejnovější z nich.

Tento zásobník se automaticky předává při volání boxů a generuje se znovu při každém uživatelem spuštěném běhu.

Vznik nových proměnných se dělí na dva typy:

4.4.1 Vstupní Argumenty boxů

Prvotní metoda vzniku nových proměnných je při volání boxu z jiného s argumenty. Tyto argumenty se předávají hodnotou a musí být navíc v přijímajícím boxu zachycené operací input, která jim i přiřadí název. Pokud proměnná zachytí méně argumentů, než bylo dodáno, zbylé se zahodí (proces vytváření proměnných postupuje zleva doprava.

Pozor, že takto zachycená proměnná může mít stejný název jako už nějaká dříve existující, v takovém případě dochází k překrytí, nikoli k editaci/mazání původní!

4.4.2 Data-box a vznik proměnných

Druhý způsob vzniku nových proměnných nastane, pokud některý z vykonávaných doit-boxů obsahuje vnořený data-box, jehož kódová sekce obsahuje číselnou hodnotu. V takovém případě se název data-boxu použije jako název proměnné s touto hodnotou.

Pozor, že takto zachycená proměnná může mít stejný název jako už nějaká dříve existující, v takovém případě dochází k překrytí, nikoli k editaci/mazání původní!

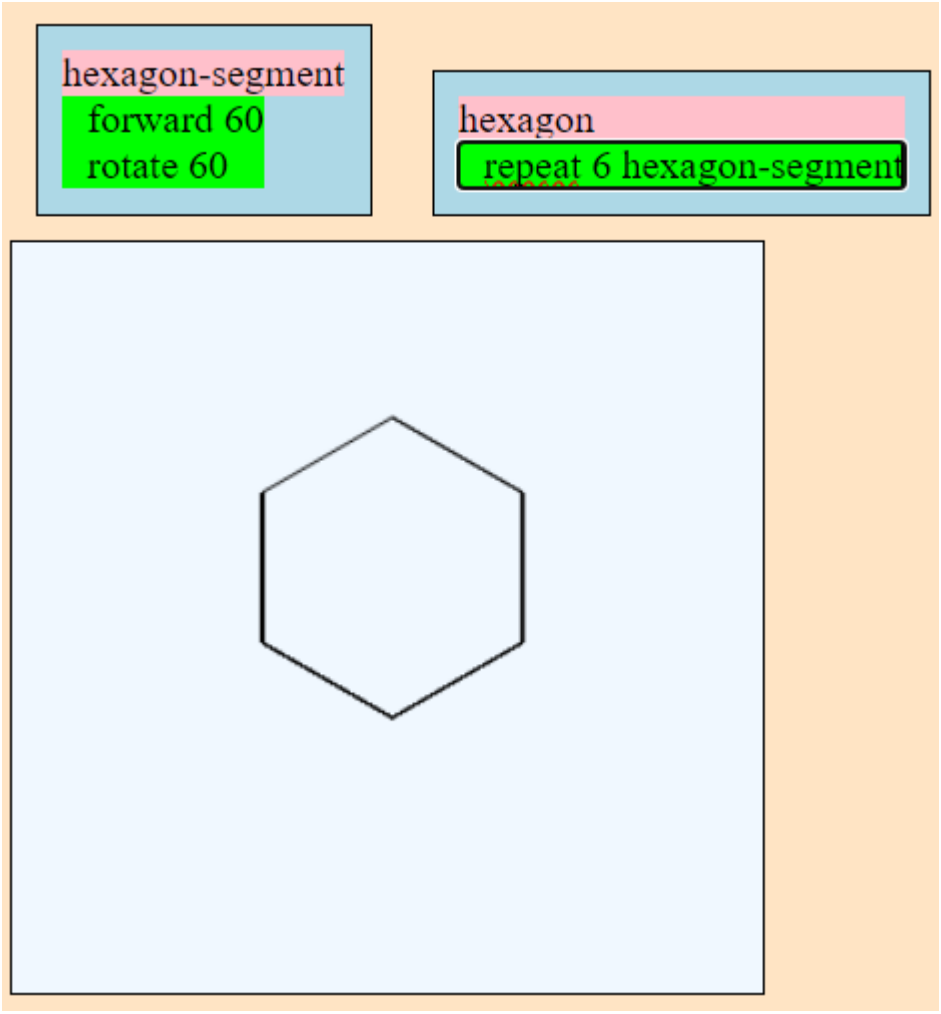
5. Obrazovky

Celý programovací systém je přístupný z hlavního okna webové stránky. Obsahuje všechny vytvořené boxy a canvas pro kreslení. Níže je vyobrazen motivační příklad vytvořeného (a později doběhnutého) programu vykreslujícího pravidelný šestiúhelník:

5.1 Obrazovka 1

Na obrázku vidíme doit-box definující kreslení jedné části šestiúhelníku a doit-box, který jej využívá k vykreslení celého obrazce. Součástí je i html canvas, ve kterém se výsledek zobrazí.

5.2 Obrazovka 2



Výsledek běhu programu po spuštění doit-boxu **hexagon** se vykreslil jako šestiúhelník.

6. Ostatní (mimofunkční) požadavky

6.1 Požadavky na výkon

Projekt nemá specifické požadavky na výkon.

6.2 Požadavky na rozšiřitelnost a začlenitelnost

Programovací systém bude v budoucnu umožňovat práci i s proměnnými textového typu a jejich editaci po vytvoření.

Programovací systém také bude v budoucnu umožňovat použití v rámci (jiných) větších webů jako „widget“, například pro dynamické vykreslení obrazců podle zadaných parametrů.

7. Negativní vymezení

Programovací systém bude testován jen pro běžné prohlížeče (využívající chromium – chrome, edge…) a Firefox.

Programovací systém aktuálně neobsahuje operace schopné pracovat s textovými proměnnými, a tudíž by se v něm tyto proměnné neměly vyskytovat.

8. Time-line & Milestones

Datum	Milník	Způsob prezentace
20.4.	Tvorba specifikace, repozitáře	osobní předvedení
10.5.	implementace prvního spustitelného dema	osobní předvedení
20.5.	implementace dalších primitives	kód v repozitory, osobní předvedení
29.5.	implementace volání jiných Boxů	kód v repozitory, osobní předvedení
8.6.	implementace jednoduchých cyklů	kód v repozitory, osobní předvedení
10.6.	implementace předávání argumentů při volání jiných boxů	kód v repozitory, osobní předvedení
17.6.	implementace proměnných – návrh prototyp	dokumentace, konzultace
27.6.	implementace proměnných – plná verze	kód v repozitory, osobní předvedení
17.9.	Finální verze specifikace	Odevzdání

Dodatek A: Vymezení pojmů

Box – skupina custom html tagů dělící se dále na doit a data-box.

Doit-box – executable varianta boxu, obsahuje box-name definující název pro budoucí referenci na box, a spustitelný box-code

Data-box – non-executable varianta boxu, obsahuje box-name a nespustitelný box-code, typicky obsahující hodnotu proměnné nebo komentář

Box-name – párový html tag sloužící k identifikaci boxu v rámci programu

Box-code – párový html tag obsahující samotný kód boxu

Primitive – jednoduché operace které nejsou box – například kreslící operace na canvas, volání jiného boxu, operace repeat (obdoba for-cyklu)

9. Poznámky

Tato specifikace je více než inspirována těmito šablonami:

- Software Requirements Specification by Karl E. Wiegers
- SAFE™ Development System Requirements