

# Průvodce programováním v HubNetu

Tento průvodce vám poskytne informace potřebné k tomu, abyste porozuměli kódům existujících aktivit HubNetu a mohli je změnit, případně si napsat své vlastní aktivity. Předpokládáme, že už aktivity umíte spustit, znáte základy programování v NetLogu a obeznámili jste se s prvky rozhraní. Více obecných informací naleznete v [Průvodci HubNetem](#).

- [Všeobecné informace o HubNetu](#)
- [Programování aktivit HubNetu](#)
  - ♦ [Nastavení](#)
  - ♦ [Příjem zpráv od klientů](#)
  - ♦ [Zasílání zpráv klientům](#)
- [HubNet na grafické kalkulačce](#)
- [HubNet na počítači](#)
  - ♦ [Jak vytvořit klientské rozhraní](#)
  - ♦ [Aktualizace zobrazení na klientech](#)
  - ♦ [Interakce v zobrazovacím okně na klientech](#)
  - ♦ [Aktualizace grafů na klientech](#)

## Všeobecné informace

Informace v této podkapitole slouží především uživatelům připojujícím se pomocí počítače, i když většina uváděného kódu může být s drobnými změnami použita i u klientů na grafické kalkulačce.

## Programování aktivit HubNetu

Hodně aktivit HubNetu sdílí kusy stejného kódu programu, např. kódu užívaného k nastavení sítě či kódu sloužícího k příjmu a vysílání informací od klientů/klientům. Rozumíte-li tomuto kódu, snadno pozměníte stávající aktivity a rychle se naučíte psát své vlastní. Začneme Vzorovým modelem (Template model), který se nachází sekci **HubNet Computer Activities -> Code Examples**. Tento

model obsahuje základní komponenty vyskytující se ve většině aktivit HubNetu. Aktivitu lze využít jako odrazový můstek pro většinu projektů.

## Nastavení

Abyste z modelu NetLogo vytvořili aktivitu HubNetu, musíte nejdříve inicializovat síť. K tomu ve většině aktivit použijete proceduru start up. start up je zvláštní procedura, již se NetLogo bude snažit spustit při otvírání jakéhokoli modelu, proto je vhodná na umístění kódu, který chcete spustit pouze jednou (bez ohledu na to, kolikrát model spustí uživatel). V HubNetu píšeme příkazy inicializující síť do této procedury, protože nám síť stačí nastavit pouze jednou. Nejdříve musíme určit druh klienta pomocí hubnet-set-client-interface; v tomto případě budeme používat počítačové klienty:

```
hubnet-set-client-interface "COMPUTER" [ ]
```

Systém inicializujete pomocí příkazu hubnet-reset, který požádá uživatele o název relace a otevře ovládací panel HubNetu (HubNet Control Center). NetLogo je připraveno naslouchat zprávám od klientů.

Když už je síť nastavena, nemusíte si dělat starosti s dalším voláním hubnet-set-client-interface nebo hubnet-reset. Podívejme se na proceduru PŘIPRAV ve vzorovém modelu:

```
to setup
  cp
  cd
  clear-output
  ask turtles
  [
    set step-size 1
    hubnet-send user-id "step-size" step-size
  ]
end
```

Většina kódu se shoduje s ostatními procedurami PŘIPRAV, ale všimněte si, že nevoláme clear-all. V tomto modelu, stejně jako ve většině aktivit nástroje HubNet v knihovně modelů, potřebujeme rod želv představující přihlášené klienty. V našem případě jsme rod nazvali students (studenti). Kdykoliv se přihlásí další klient, vytvoříme studenta a zaznamenáme všechny informace, jež by se nám mohly později hodit, do proměnné želvy. A protože nechceme, aby se uživatelé odhlašovali a znovu přihlašovali pokaždé, když nastavujeme aktivitu, nemůžeme zrušit všechny želvy, ale pouze nastavíme všechny proměnné na výchozí hodnoty a uvědomíme klienty o změnách, které provádíme (viz dále).

## Příjem informací od klientů

V průběhu aktivity se budou přenášet data mezi klienty HubNetu a serverem. Většina aktivit volá ve smyčce `go` (START) proceduru, která ověřuje, zda nedošly od klientů nové zprávy. Tato procedura se nazývá `listen-clients`:

```
to listen-clients
  while [ hubnet-message-waiting? ]
  [
    hubnet-fetch-message
    ifelse hubnet-enter-message?
    [ create-new-student ]
    [
      ifelse hubnet-exit-message?
      [ remove-student ]
      [ execute-command hubnet-message-tag ]
    ]
  ]
end
```

Dokud jsou ve frontě nějaké zprávy, tato smyčka je po jedné zachytí. `hubnet-fetch-message` změní další zprávu ve frontě na aktuální a nastaví reportéry `hubnet-message-source`, `hubnet-message-tag` a `hubnet-message` na odpovídající hodnoty. Klienti posílají zprávu při každém přihlášení a odhlášení uživatele a při každém použití prvku rozhraní, tj. když uživatel stiskne tlačítko, posune posuvník, klikne do zobrazení atd. Procházíme každou zprávu a rozhodujeme, jakou akci provedeme v závislosti na povaze zprávy (přihlášení, odhlášení či jiná) či na hodnotách proměnných `hubnet-message-tag` (název prvku rozhraní) nebo `hubnet-message-source` (jméno klienta, od kterého zpráva přišla).

U přihlášení vytvoříme želvu s identifikačním jménem uživatele `user-id`, které se shoduje s `hubnet-message-source`, tj. se jménem, pod kterým uživatel vstupuje do aktivity. Toto jméno musí být unikátní.

```
to create-new-student
  create-students 1
  [
    set user-id hubnet-message-source
    set label user-id
    set step-size 1
    send-info-to-clients
  ]
end
```

V tuto chvíli nastavíme ostatní proměnné klienta na výchozí hodnoty a pošleme je klientům, je-li třeba. Pro každý odpovídající prvek rozhraní klienta, který si udržuje stav, tj. pro vše, co bude na serveru

globální proměnnou – posuvníky, roletky, přepínače a vstupní pole, deklarujeme vlastní proměnnou students-own. Tyto proměnné musejí být synchronizovány s hodnotami viditelnými na klientovi.

Když se klienti odhlašují, pošlou odhlašovací zprávu serveru, což umožňuje vymazat všechny informace, jež o nich byly uloženy. V tomto případě jednoduše řekneme dané želvě, aby umřela:

```
to remove-student
  ask students with [user-id = hubnet-message-source]
  [ die ]
end
```

Všechny ostatní zprávy jsou prvky rozhraní identifikované pomocí hubnet-message-tag, což je název objevující se v rozhraní klienta. Pokaždé, když se prvek rozhraní změní, odešle se serveru zpráva. Pokud stavy aktuálně zobrazené v rozhraní klienta neuložíte, nebudou přístupné ostatním částem modelu. Z toho důvodu jsme pro každý prvek vyjadřující stav (posuvníky, přepínače atd.) deklarovali proměnné students-own. Když od klienta dostaneme zprávu, nastavíme proměnnou želvy podle obsahu zprávy:

```
if hubnet-message-tag = "step-size"
[
  ask students with [user-id = hubnet-message-source]
  [ set step-size hubnet-message ]
]
```

K tlačítkům však žádná data přiřazena nejsou, takže jim neodpovídá ani žádná proměnná želvy. Místo toho oznamují akci vykonanou klientem – stejně jako v případě běžného tlačítka, i zde ke každému bývá asociována procedura, kterou voláte, kdykoliv obdržíte zprávu o stisku tlačítka. I když to není zcela nutné, bývá procedura zpravidla procedurou želvy, tj. něčím, co želva-student asociovaná se zdrojem vzkazu může vykonat:

```
if command = "move left"
[ set heading 270
  fd 1 ]
```

## Zasílání zpráv klientům

Jak jsme již zmiňovali, můžete také zasílat hodnoty jakémukoliv prvku rozhraní, jenž umí zobrazit informaci: ukazatele, posuvníky, přepínače, roletky a vstupní pole (grafy a zobrazení jsou zvláštními případy, jimž se budeme věnovat zvlášť). Pro zasílání informací slouží dvě primitiva – hubnet-send a hubnet-broadcast. Příkaz hubnet-broadcast zašle informaci všem klientům a hubnet-send konkrétní skupině či vybraným klientům. Nic se však na klientském počítači neaktualizuje automaticky, pokud se změní nějaká hodnota na serveru, musíte aktualizovat ukazatele na klientovi. Jestliže na serveru změníte proměnnou asociovanou s dalším prvkem rozhraní, aniž byste tak reagovali na zprávu

o změně od toho daného prvku, musíte rovněž aktualizovat hodnotu na klientovi. Řekněme například, že máme na klientovi posuvník „step-size“ a ukazatel „Step Size“ (všimněte si, že názvy musejí být rozdílné), aktualizujeme je následujícím kódem:

```
if hubnet-message-tag = "step-size"
[
  ask student with [ user-id = hubnet-message-source ]
  [
    set step-size hubnet-message
    hubnet-send user-id "Step Size" step-size
  ]
]
```

Zasílat můžete jakékoliv typy dat – čísla, řetězce, seznamy, seznamy seznamů, seznamy řetězců, avšak v případě, že data nejsou použitelná pro přijímací prvek rozhraní (když např. zašlete řetězec posuvníku), bude zpráva ignorována. Následuje několik ukázek kódu pro rozdílné typy dat:

typ dat	příklad hubnet-broadcast	příklad hubnet-send
čísla	hubnet-broadcast "A" 3.14	hubnet-send "jimmy" "A" 3.14
řetězec	hubnet-broadcast "STR1" "HI THERE"	hubnet-send ["12" "15"] "STR1" "HI THERE"
seznam čísel	hubnet-broadcast "L2" [1 2 3]	hubnet-send hubnet-message-source "L2" [1 2 3]
matice čísel	hubnet-broadcast "[A]" [[1 2] [3 4]]	hubnet-send "susie" "[A]" [[1 2] [3 4]]
seznam řetězců (pouze pro počítačové klienty HubNetu)	hubnet-broadcast "user-names" [{"jimmy" "susie"} [{"bob" "george"}]	hubnet-send "teacher" "user-names" [{"jimmy" "susie"} ["bob" "george"]]

## Příklady

Projděte si modely v **Models Library**, složka **HubNet Computer Activities** a **HubNet Calculator Activities** a v okně procedur sledujte, jak jsou výše uvedená primitiva použita v praxi. Začněte modelem Nemoc (Disease).

## HubNet na grafické kalkulačce

Pro více informací o psaní aktivit HubNetu pro klienty na grafických kalkulačkách nás kontaktuje na Inquire Learning, [calc-hubnet@inquirelearning.com](mailto:calc-hubnet@inquirelearning.com), nebo navštivte <http://www.inquirelearning.com/calc-hubnet.html>.

## HubNet na počítači

Následující informace se týkají použití HubNetu na počítači.

### Jak vytvořit klientské rozhraní

Otevřete editor klienta HubNetu (HubNet Client Editor), který se nachází v menu **Tools**. Přidejte tlačítka, posuvníky, přepínače, ukazatele, grafy, roletky či poznámky stejným způsobem jako v panelu **Interface**. Všimněte si, že vkládaná informace se u každého prvku malinko liší od panelu Interface. Prvky rozhraní na klientu interagují s modelem rozdílně, místo přímého odkazu na příkazy a reportéry posílají zprávy zpět na server a teprve model potom rozhodne, jak ho tyto zprávy ovlivní. Všechny prvky rozhraní na klientu mají tag, což je unikátní název identifikující daný prvek. Když server od prvku obdrží zprávu, objeví se tag v `hubnet-message-tag`.

Když máte například tlačítko nazvané „move left“, posuvník „step-size“, přepínač „all-in-one-step?“ a ukazatel „Location:“, budou tagy pro prvky rozhraní následující:

prvek rozhraní	tag
move left	move left
step-size	step-size
all-in-one-step?	all-in-one-step?
Location:	Location:

Všimněte si, že můžete mít pouze **jeden** prvek rozhraní s daným názvem. Kdyby v rozhraní klienta bylo více prvků se stejným tagem, začal by se model chovat nevypočitatelně, protože by nebylo jasné, kterému prvku byla informace adresována.

## Aktualizace zobrazení na klientech

Nejjednodušší způsob, jak zobrazit svět v zobrazovacím okně klienta, je použít zrcadlové zobracení. Při zrcadlovém zobrazení (zapíná se v ovládacím panelu HubNetu) se zobrazení na klientu automaticky aktualizuje, aby odrazilo stav světa. Všichni klienti mají stejné zobrazení, jež se shoduje se zobrazením na serveru. Zobrazení se aktualizuje přibližně pětikrát za sekundu, což znamená poměrně velké množství zpráv zasílaných klientům. Pokud máte s efektivitou modelu problémy, použijte příkazy `no-display` či `display` a snižte počet aktualizací zasílaných klientům.

Chcete-li mít nad aktualizacemi zobrazení více kontroly, použijte `hubnet-broadcast-view` a `hubnet-send-view` a explicitně zašlete nastavení buď všem klientům, nebo jen určeným v tomto pořadí. Upozorňujeme vás však, že zrcadlové zobrazení aktualizuje pouze informace, jež se změnily od poslední aktualizace, kdežto příkazy `hubnet-broadcast-view` a `hubnet-send-view` zasílají pokaždé kompletní informace o světě. Výsledkem je, že zprávy těchto příkazů jsou mnohem větší a aktualizace tudíž probíhá pomaleji.

Pokud na klientech není žádné zobrazení či pokud nejsou v zaškrťovacím rámečku Mirror 2D View v ovládacím panelu HubNetu (Zrcadlové 2D zobrazení) zaškrtnuti Clients (klienti), nezasílají se klientům žádné zprávy týkající se zobrazení.

**Upozornění:** Jelikož jsou příkazy `hubnet-broadcast-view` a `hubnet-send-view` experimentální primitiva, může se jejich chování v budoucích verzích NetLoga změnit.

Poznámka: Některé z vlastností zobrazení v NetLogu, např. zacyklení světa (View Wrapping) a perspektiva pozorovatele (Observer Perspectives), ještě nejsou implementovány na klientech HubNetu.

## Interakce v zobrazovacím okně na klientech

Pokud je u klienta zobrazení (View), pošle se serveru zpráva pokaždé, kdy uživatel do zobrazení klikne. Zpráva má tag „View“ a skládá se ze seznamu o dvou položkách obsahující souřadnice X a Y. Chceme-li např. zbarvit všechna políčka, na které klient kliknul, červeně, použijeme následující kód NetLoga:

```
if hubnet-message-tag = "View"
[
  ask patches with [ pxcor = (round item 0 hubnet-message) and
                    pycor = (round item 1 hubnet-message) ]
  [ set pcolor red ]
]
```

## Aktualizace grafů na klientech

Jestliže máme spuštěno zrcadlové zobrazení (v ovládacím panelu HubNetu) a v modelu NetLoga se změní graf, jehož název existuje i na klientech, je klientům poslána zpráva, na jejímž základě provede graf stejnou změnu. Řekněme si například, že v nějakém modelu HubNetu se v NetLogu i na klientech vyskytuje graf nazvaný Dodávka mléka (Milk Supply). Tento graf je v NetLogu aktuální, takže stačí do příkazového panelu napsat:

```
plot 5
```

To způsobí, že je všem klientům zaslána zpráva, jež jim sděluje, že musejí zakreslit bod v hodnotě 5 na ose Y do další pozice grafu. Pozor, v případě, že v jeden okamžik hodně zakreslujete, obdrží klienti velké množství zpráv.



Copyright 1999-2009 by Uri Wilensky.  
Všechna práva vyhrazena.

Aplikace NetLogo, modely i dokumentace jsou šířeny veřejnosti zdarma pro účel tvorby a studia modelů. Software, modely a dokumentaci je možné pro studijní a výzkumné účely používat a měnit, a to za podmínky, že je výsledný produkt nabízen bezplatně a s uvedením informace o autorských právech a jménem původce na všech kopiích a související dokumentaci.

Pro jiné využití - než jsou výše zmíněné nekomerční způsoby - celku i jednotlivých částí (a to jak v původní, nebo změněné podobě) je třeba předem požádat o svolení od Uri Wilensky. Software, modely ani dokumentace nesmějí být užívány, přepisovány, ani upravovány jako součást komerčního softwaru nebo hardwaru bez předchozího získání licence od Uri Wilensky. Nezaručujeme kompatibilitu tohoto systému s jakýmkoliv jiným systémem a neposkytujeme žádné záruky.

Pro účely citování v akademických publikacích používejte tento odkaz:  
Wilensky, U. (1999). NetLogo. <http://ccl.northwestern.edu/netlogo>. Center for Connected Learning and Computer-Based Modeling. Northwestern University, Evanston, IL.