Oprogramowanie do tworzenia eksperymentów ekonomicznych on-line

Paweł Kowal, absolwent WNE UW, programista, **Tomasz Kopczewski**, adiunkt na WNE UW, **Robert Borowski**, absolwent WNE UW, doktorant SGH

Ekonomia eksperymentalna to gałąź ekonomii zajmująca się analizą wybranych teorii ekonomicznych w ściśle kontrolowanych warunkach laboratoryjnych. Za pomocą odpowiednio zaplanowanych eksperymentów można sprawdzić, czy teoretycznie przewidziane zachowania ekonomiczne będą występowały w rzeczywistości. W czasach poprzedzających rozwój ekonomii eksperymentalnej uważano, że eksperymenty są nieprzydatne w ekonomii, ponieważ zajmuje się ona dużymi, skomplikowanymi systemami występującymi w naturze [Mill J. S., 1836 (1967), s. 124]. Takich systemów nie dało się odtworzyć w laboratorium. Obecnie ekonomia eksperymentalna jest jednak jednym z ważnych narzędzi ekonomicznych [Kagel, Roth, 1995]. Początkowy impuls do zainteresowania ekonomią eksperymentalną pojawił się w trakcie prac nad teorią indywidualnego wyboru. Pozycja ekonomii eksperymentalnej umocniła się podczas badań nad mikroekonomicznymi teoriami, bazującymi na indywidualnych preferencjach, które trudno obserwuje się w środowisku naturalnym. Powstanie teorii oczekiwanej użyteczności i teorii gier spowodowało stałe stosowanie metod eksperymentalnych w ekonomii [Kahneman, Tversky, 2000]. Wykorzystywane jeszcze kilka lat temu, przed rozkwitem technologii komputerowych, papierowe metody prowadzenia doświadczeń są bardzo pracochłonne i pozwalają na popełnienie błędów, które nawet jeśli nie dyskredytują pod względem metodologicznym wyników, to na pewno mogą wnosić spore obciążenie. Dostępne programy komputerowe okazały się trudne do użycia bądź nie miały możliwości wymaganych do przeprowadzenia specyficznych eksperymentów.

1. Efektywność oprogramowania

Można się zastanawiać, czy każde oprogramowanie, zwłaszcza wspierające eksperymenty ekonomiczne, jest efektywne. Jako efektywność należy rozumieć porównanie kosztów stworzenia i korzyści z zastosowania oprogramowania. Koszty to oczywiście czas poświęcony na oprogramowanie i przygotowanie interfejsu użytkownika do pojedynczego eksperymentu. Na korzyści

składają się wiarygodność wyników, czas przetwarzania danych, ale także korzyści skali — możliwość wykorzystania kodu programu w tworzeniu innego eksperymentu.

Można przyjąć kilka wyróżników efektywnego oprogramowania. Są to m.in.:

— możliwość szybkiego tworzenia pojedynczych eksperymentów,
— większe możliwości środowiska — uniwersalność środowiska — wykorzystywanie części kodów w szerokiej gamie eksperymentów, bez skupiania się na wybranym typie doświadczenia, możliwość zbierania doświadczeń dzięki dobrej dokumentacji, zrozumiałej dla użytkowników (np. w języku ojczystym),
— wspomaganie procesu zbierania danych w celu łatwiejszej ich obróbki,
— możliwości wykorzystania wiedzy zdobytej przy planowaniu doświadczenia w innych dziedzinach, np. programowaniu komputerowym.

Efektywność oprogramowania można badać w kontekście dwóch konkurencyjnych programów do tworzenia eksperymentów, ale można także patrzeć przez pryzmat eksperymentów tradycyjnych, przeprowadzanych w sali z użyciem kartki i długopisu. W takim porównaniu eksperyment tradycyjny okazuje się zazwyczaj lepszy od komputerowego w fazie projektowania, polegającej na ustaleniu testowanej hipotezy, prawdopodobnych wyników, reguł rządzących doświadczeniem i sposobu zbierania danych. Projektowanie formularzy, bazy danych wyników oraz kolejnych etapów gry jest prostsze i mniej czasochłonne przy użyciu tradycyjnych narzędzi, gdyż liczy się tu przede wszystkim doświadczenie eksperymentalne, umiejętności programistyczne zaś należy traktować jako drugoplanowe.

Forma komputerowa zdobywa często przewagę nad formą tradycyjną w kolejnej fazie procesu przeprowadzania eksperymentu, czyli w eksperymencie właściwym. Dzieje się tak, gdy stworzenie kwestionariuszy jest czasochłonne, ponieważ jest ich dużo lub są one kosztowne albo niemożliwe do stworzenia przed eksperymentem, bo ich treść zależy od przebiegu eksperymentu. Sam przebieg eksperymentu jest też łatwiejszy do opanowania, gdy wykorzystywane są narzędzia informatyczne. Typowy eksperyment jest podzielony na tury. Każda tura jest zakończona zebraniem wyników i ewentualnie stworzeniem danych początkowych do następnej tury na podstawie tych wyników. Zebranie kwestionariuszy zajmuje więcej czasu niż przesłanie wyników między komputerami. Tradycyjna forma eksperymentu jest podatna na powstawanie błędów. Dane są przetwarzane w trakcie eksperymentu, kiedy czas jest ograniczony, więc często rezygnuje się z kontroli wyników. Po drugie, dane na kwestionariuszu mogą być nieczytelne, co spowoduje ich złą interpretację. Komputer gwarantuje, że dane będą jednakowe dla badanego i prowadzącego, a ich obróbka daje zawsze jednakowy wynik. Poza tym wyniki te są dzięki temu dostępne natychmiast. W ten sposób prowadzący jest w stanie przeprowadzić więcej iteracji. Większa liczba tur to bardziej „pewne" wyniki, ale też mniejsze zmęczenie wśród badanych. Eksperyment komputerowy pozwala także na większą kontrolę anonimowości. W zależności od

doświadczenia prowadzący może zdecydować, czy gracze poznają swoją tożsamość, czy też nie. Anonimowość odgrywa w wielu eksperymentach bardzo ważną rolę, np. w doświadczeniu dotyczącym kartelu gracz anonimowy będzie prawdopodobnie częściej łamał zmowę cenową niż gracz nieanonimowy.

Duże znaczenie w eksperymentach z użyciem komputerów ma wykorzystanie internetu. Można założyć, że w tradycyjnej formie eksperymentu jego zasięg jest ograniczony do jednego pomieszczenia. Problem ten można częściowo rozwiązać przez prowadzenie eksperymentów za pomocą np. telefonu komórkowego [Michałek, 2007], jednak w tym wypadku problematyczne staje się choćby wysyłanie i zbieranie kwestionariuszy. Poza tym prowadzący nie ma wtedy kontroli nad prawdziwością danych. W przypadku eksperymentu komputerowego, opartego na aplikacji internetowej, odległości nie grają roli. Osoby badane nie muszą zbierać się w jednym miejscu, mogą wziąć udział w eksperymencie nawet z domu.

Jeszcze innym aspektem wykorzystania komputerów jest skala przeprowadzanych eksperymentów. Informatyzacja pozwala na uzyskanie znacznie większej liczby wyników, niż w eksperymencie tradycyjnym, ograniczającym zwykle liczbę uczestników ze względu na pojemność sali i czas przetwarzania formularzy. Komputery pozwalają na eksperymenty, w których udział bierze kilkaset czy kilka tysięcy osób. Poza tym eksperyment komputerowy jest w stanie zapewnić ciekawszy przekaz. Prowadzący nie musi się ograniczać do tego, co może zawierać kartka papieru. Może skorzystać w eksperymencie z dźwięku czy filmu. Tradycyjna forma również ma takie możliwości, jednak „multimedialny" eksperyment w tradycyjnej formie jest znacznie droższy.

Także obróbka formularzy po zakończeniu eksperymentów jest łatwiejsza. Zbieranie danych jest wygodniejsze w formie komputerowej. Pozwala ono na uniknięcie żmudnego przepisywania danych z kwestionariuszy do komputera, a przy okazji pozwala uniknąć błędów przy przepisywaniu. Dobrze zaplanowany format wyników będzie miał najprawdopodobniej postać tabelaryczną, warto żeby nie wymagał specjalnych narzędzi do obróbki. Prosty plik tekstowy, z polami oddzielonymi znakiem tabulacji bądź innym neutralnym znakiem może być odczytany za pomocą dowolnego edytora tekstu, a jednocześnie bardzo łatwo wczytać go do arkusza kalkulacyjnego. Jeśli arkusz kalkulacyjny jest narzędziem niewystarczającym, to taka forma danych jest akceptowalna przez popularne programy do analizy statystycznej czy też systemy bazodanowe.

2. Dostępne oprogramowanie do tworzenia eksperymentów

Oprogramowanie do tworzenia eksperymentów ekonomicznych jest bardzo różnorodne. Można je klasyfikować wg różnych kryteriów. Są to m.in. interfejs użytkownika, specjalizacja programów, środowisko działania czy język programowania. Istnieją interfejsy użytkownika graficzne i tekstowe, przy czym te pierwsze mają znaczną przewagę nad drugimi. Dzięki ich zastosowaniu w doświadczeniu można wykorzystać obrazy i filmy, co jest niemożli-

we w interfejsach tekstowych. Oprogramowanie może być bardzo wyspecjalizowane, czyli pisane dla konkretnego rodzaju eksperymentu, z dokładnie ustalonym algorytmem przebiegu i pozwalające przeprowadzić jeden rodzaj eksperymentu, np. gry w postaci macierzowej testujące równowagę Nasha. Można także spotkać środowiska uniwersalne, za których pomocą można przeprowadzić szeroką gamę eksperymentów. Uwzględniając środowisko działania można wyróżnić aplikacje „webowe" i dedykowane programy. Aplikacje „webowe" to takie rozwiązania, które w swej pracy korzystają z serwera www. Potencjalny użytkownik korzysta z nich używając przeglądarki www. Programy dedykowane to rozwiązania, które wymagają od badanego, aby w swoim komputerze zainstalował specjalny program. Wzięcie udziału w eksperymencie polega w tym wypadku na uruchomieniu tego programu. Oprogramowanie różni się też ze względu na język programowania eksperymentów. Można wyróżnić trzy grup środowisk. Pierwsza z nich to grupa programów, które programowane są za pomocą plików konfiguracyjnych. W tej grupie przebieg eksperymentu jest ustalony, jedynie parametry eksperymentu, takie jak liczba rund czy punktacja poszczególnych graczy mogą ulec zmianie. Druga grupa środowisk to programy, programowane za pomocą dedykowanego języka programowania. Środowisko zawiera w sobie język programowania, w którym można ustalić dane dotyczące zarówno przebiegu eksperymentu, jak i jego parametry. Bardzo często języki tego typu to języki graficzne. Ostatnia grupa w tym podziale to środowiska wykorzystujące w swej pracy jeden z „prawdziwych" języków programowania. Na ogół środowiska takie to biblioteki pewnych funkcji, które wykorzystuje się przy pisaniu danego eksperymentu.

Ośrodki akademickie na całym świecie tworzą oprogramowanie do konstrukcji eksperymentów, niestety często tylko na swoje potrzeby. Jednak pomimo ich różnorodności, trudno znaleźć ideał. Oto krótki opis kilku z dostępnych środowisk.

ESL (Economic Science Laboratory) to laboratorium na uniwersytecie w Arizonie[1]. Laboratorium wykorzystuje w swojej pracy zbiór programów (stworzonych dla własnych potrzeb) dedykowanych do różnych rodzajów eksperymentów. Dostępne rodzaje eksperymentów to: tradycyjne gry z tabelą „wypłat", symulacja rynku, symulacja giełdy i symulacja aukcji (jej różne rodzaje). Programy te to aplikacje napisane w języku Java, dzięki czemu można ich używać w różnych systemach operacyjnych. Programy te należy najpierw zainstalować (wystarczy jeden, w zależności od badanej hipotezy) na każdym z używanych komputerów, pomocą może być wersja instalacyjna działająca przez stronę www. Ponieważ każdy rodzaj eksperymentu to oddzielna aplikacja, programowanie eksperymentów polega na wpisaniu w odpowiedni formularz po stronie serwera odpowiednich parametrów doświadczenia, np. tabeli wypłat. Programy mają graficzny interfejs, który nie jest

[1] http://www.econlab.arizona.edu/.

jednak modyfikowalny. Serwer pozwala na odzyskiwanie zerwanych połączeń. Dane na temat każdej gry są zapisywane w pliku tekstowym, plik ma ściśle określoną strukturę w zależności od badanego problemu.

CASSEL (California Social Science Experimental Laboratory) to wspólny projekt kilku amerykańskich organizacji[2]. Laboratorium składa się z 70 komputerów (w przyszłości ma ich być około 120). Zostało zbudowane za około 2 mln USD specjalnie w celu prowadzenia eksperymentów ekonomicznych. Mogą z niego korzystać ekonomiści, socjologowie i psychologowie. Laboratorium wykorzystuje stworzone przez siebie oprogramowanie. Programy używane w CASSEL to rozwiązania specjalizowane, każdy z programów odpowiada za badanie innej hipotezy. Dostępne moduły to gry z tabelą „wypłat" dla dwóch i więcej graczy, moduł symulacji rynku i aukcji oraz tzw. moduł *free form*, który pozwala na dowolne ustalenie przebiegu gry. Cześć modułów jest programowana za pomocą środowiska graficznego, w ten sposób ustalane są np. tabele „wypłat". Pozostałe moduły są programowane za pomocą dedykowanego hybrydowego języka. W języku tym najpierw graficznie tworzy się graf, który opisuje algorytm doświadczenia, potem odpowiednie wierzchołki grafu są wypełniane kodem. Kod ten może zawierać definicje zmiennych, operacje na tych zmiennych i wyrażenia warunkowe. Programy używane przez CASSEL są napisane w Javie, nie wymagają instalacji na komputerze, gdyż są dostarczane w formie strony www.

RatImage to środowisko używane w laboratorium ekonomii eksperymentalnej na uniwersytecie w Bonn. Jest to właściwie biblioteka napisana w języku Pascal. Tworzenie eksperymentu za pomocą tego środowiska polega na pisaniu programu w języku Borland Turbo Pascal. W ten sposób możliwości prowadzącego eksperyment nie są ograniczone i może on badać dowolną hipotezę. Całe środowisko jest trochę przestarzałe. Interfejs, co prawda graficzny, ma niewielkie możliwości. Program jest kompilowany do kodu maszynowego, co powoduje, że może być uruchamiany jedynie w środowisku Windows na komputerach klasy PC. Aby skorzystać z programu należy go najpierw zainstalować. Środowisko nie udostępnia serwera, gracze łączą się ze sobą na zasadzie *point-to-point*, czyli każdy gracz z każdym (w zależności od tego z kim chce zagrać). Główną (właściwie jedyną) zaletą środowiska jest możliwość programowania eksperymentów w języku Pascal, który jest nadal dość popularny i łatwy do nauczenia.

z-Tree to środowisko stworzone na uniwersytecie w Zurychu. Program może być używany do takich klas eksperymentów, jak dobra publiczne, symulacja rynku czy symulacje różnych rodzajów aukcji [Fischbacher, 2007]. Program działa w środowisku Windows, a dokładniej współpracuje z nowszymi wersjami takimi jak NT, 2000 i XP. Starsze wersje Windows powodują, że program często się zawiesza. Podobno można uruchomić środowisko także w systemie Linux, jednak wymaga to dodatkowych, skomplikowanych operacji.

[2] http://research.cassel.ucla.edu/software.htm.

Program jest ciągle ulepszany i ma dość aktywną listę mailingową użytkowników. Do tworzenia eksperymentów wykorzystano dedykowany język, część jest graficzna, część tekstowa. Środowisko jest zbudowane w architekturze klient-serwer. Aplikacje klienta trzeba przed użyciem zainstalować na komputerach. Tworzone eksperymenty mają graficzny interfejs, jednak jest on ograniczony, nie można np. używać obrazów.

Laboratorium ekonomii eksperymentalnej na uniwersytecie w Mannheim ma całkowicie odmienne od wcześniej wymienionych podejście[3]. W prowadzonych eksperymentach nie używa żadnego specjalizowanego oprogramowania. Eksperymenty są pisane jako dynamiczne strony www w PHP. W ten sposób nie ma żadnych ograniczeń dotyczących logiki eksperymentu. Takie podejście ma jednak pewne wady. Pisanie dynamicznych stron www w języku PHP jest pracochłonne, zwłaszcza ich testowanie zajmuje dużo czasu. PHP to język powszechnie używany, jednak powoli staje się przestarzały. Pisanie stron w tym języku jest trudne jednak nauczenie się jego podstaw wystarczy do prostych eksperymentów, a zdobyta w ten sposób wiedza może być użyta w innych dziedzinach.

Przedstawione powyżej programy nie spełniają jednocześnie dwóch podstawowych założeń efektywnego programowania: możliwość szybkiego tworzenia pojedynczych eksperymentów oraz uniwersalności środowiska programistycznego. Najbliższy ideału jest program z-Tree. Został on stworzony w celu szybkiego tworzenia eksperymentów przez osoby bez doświadczenia programistycznego. Jednak wykorzystanie specyficznego języka programowania, stworzonego tylko na potrzeby tego programu oraz zamknięty kod spowodowały, że program ten sprawdza się jedynie w sieci lokalnej działającej w środowisku Windows. Krańcowo różne rozwiązanie, czyli tworzenie eksperymentów w języku PHP, powoduje znaczne zwiększenie kosztów implementacji pojedynczego eksperymentu.

3. LabSEE — odpowiedź na nieefektywność istniejącego oprogramowania

W odpowiedzi na istniejącą wciąż niszę w oprogramowaniu do tworzenia eksperymentów on-line w ramach Laboratorium Ekonomii Eksperymentalnej powstaje program (środowisko) do przeprowadzania eksperymentów ekonomicznych LabSEE. Jest to połączenie efektywności z-Tree oraz uniwersalności środowiska programistycznego. Program ten w pełni wykorzystuje „premię za zacofanie". Większość programów do tworzenia eksperymentów powstała w czasach, gdy programowanie sieciowe było dopiero tworzone. Obecnie rozwiązania sieciowe są już na tyle dojrzałe, że nie należy spodziewać się zmiany kierunku ich rozwoju. Jako środowisko programistyczne została wybrana Java.

[3] http://www.kirchkamp.de/.

Obecnie używanych jest kilkanaście uniwersalnych języków programowania. Można je podzielić na dwie klasy. Pierwsza z nich to języki strukturalne. Jest to grupa starszych języków takich jak Pascal czy C. Drugą grupą języków są języki obiektowe. Do tych języków można zaliczyć C++, Javę czy C#.

Języki obiektowe powoli wypierają strukturalne. Dlatego wybór był przeprowadzony wśród tych języków. C# jest najmłodszym z wymienionych[4]. Jest to język zaprojektowany przez firmę Microsoft, pojawił się na rynku w 2000 r. C# bardzo przypomina Javę, z której usunięto kilka elementów. C# działa, współpracując z tzw. Microsoft .NET Framework. Samo środowisko .NET Framework nie jest związane z żadnym konkretnym językiem programowania, działa np. z C++, Visual Basic, J\#. Zadaniem .NET Framework jest zarządzanie różnymi elementami systemu, których obsługa do tej pory zajmowała programistom najwięcej czasu. Dzięki temu twórcy oprogramowania mają skupić się na dostarczaniu funkcjonalności, nie tracąc czasu np. na zarządzanie pamięcią lub obsługę komunikacji procesów działających w środowisku rozproszonym. Podstawowym elementem.NET jest CLR (*Common Language Runtime*). Jest to nic innego niż interpreter kodu kompilowanego na platformę .NET.

Kod kompilowany dla.NET nie jest kodem żadnej platformy sprzętowej. Podobnie jak w Javie jest on interpretowany na bieżąco w trakcie wykonywania. Ze względu na młody wiek C# nie ma jeszcze takiego wsparcia jak inne wymienione języki. Dużo materiałów udostępnia sam producent, firma Microsoft. Jednak fakt, że większość dostępnej wiedzy pochodzi bezpośrednio od twórcy języka oznacza, że język ten nie jest tak popularny jak pozostałe. Niewielka liczba dostępnych kompilatorów i środowisk programistycznych pozostawia go na razie w tyle za innymi językami.

Początki C++ sięgają 1980 roku, kiedy to jego twórca, Bjarne Stroustrup zaczął pracę nad „C z obiektami". W 1982 roku Stroustrup postanowił poprawić swoje dzieło i w ten sposób pierwsza wersja C++ ujrzała światło dzienne w 1983 roku. Wzmożona praca nad tym językiem trwała do 1985 roku, kiedy stworzona została pierwsza wersja komercyjna. Praca nad językiem nie zakończyła się i niektóre z jego możliwości zostały dodane później. Prace nad standardem ISO języka C++ zostały zakończone w 1998 roku.

Język C++ to najczęściej używany na rynku język programowania obiektowego. Mimo popularności ma on kilka wad. Po pierwsze brakuje w nim bibliotek standardowych. Te, które są, to tylko drobny wycinek tego co proponują języki C# czy Java. C++ powstał z języka C. W ten sposób do języka zostały dodane funkcje, które powodują, że język ten nie jest tak bezpieczny i łatwy w programowaniu jak pozostałe. Bardzo błędogenne jest użycie wskaźników. Inną wadą jest fakt używania destruktorów. W przeciwieństwie do Javy czy C# jest możliwe stworzenie programów, które będą miały trudne do wykrycia błędy powodujące tzw. wycieki pamięci. C++ w swojej składni posiada też

[4] http://msdn.microsoft.com/.

kilka konstrukcji, które czynią go mniej zrozumiałym niż Java. Składnia C++ powoduje, że jest on mniej czytelny od wymienionych języków, a co gorsza programy w nim napisane mogą nie być zgodne z przyszłymi wersjami.

Java to twór firmy Sun [Stroustrup, 2002]. Prace nad językiem rozpoczęły się w 1991 roku, kiedy okazało się, że C++ nie jest dla firmy Sun wystarczająco dobry. W 1993 r. język, nad którym pracowano, został przemianowany z Oak na Java. Wypuszczono na rynek kilka rozwiązań wykorzystujących Javę, ale nie odniosły one większego sukcesu. W tym czasie www rozwijało się bardzo dynamicznie i firma Sun zauważyła możliwości języka niezależnego od platformy. W 1995 została wypuszczona wersja 1.0. Rok później pojawiła się wersja 1.1. Obecnie ostatnim standardem jest 1.6.7 i trwają pracę nad kolejną wersją.

Java to język niezależny od platformy sprzętowej i systemu operacyjnego. Kod pisany w Javie jest kompilowany do tzw. kodu pośredniego. Następnie w zależności od platformy w trakcie wykonywania kod ten jest kompilowany do kodu procesora przez tzw. wirtualną maszynę Javy (*Java Virtual Machine*). Takie podejście powoduje, że język jest bardzo przenośny, istnieją maszyny wirtualne na wszystkie liczące się systemy operacyjne. Co więcej kod Java wykonywany na różnych maszynach będzie się zachowywał jednakowo. Np. interfejs użytkownika będzie jednakowy bez względu na system operacyjny.

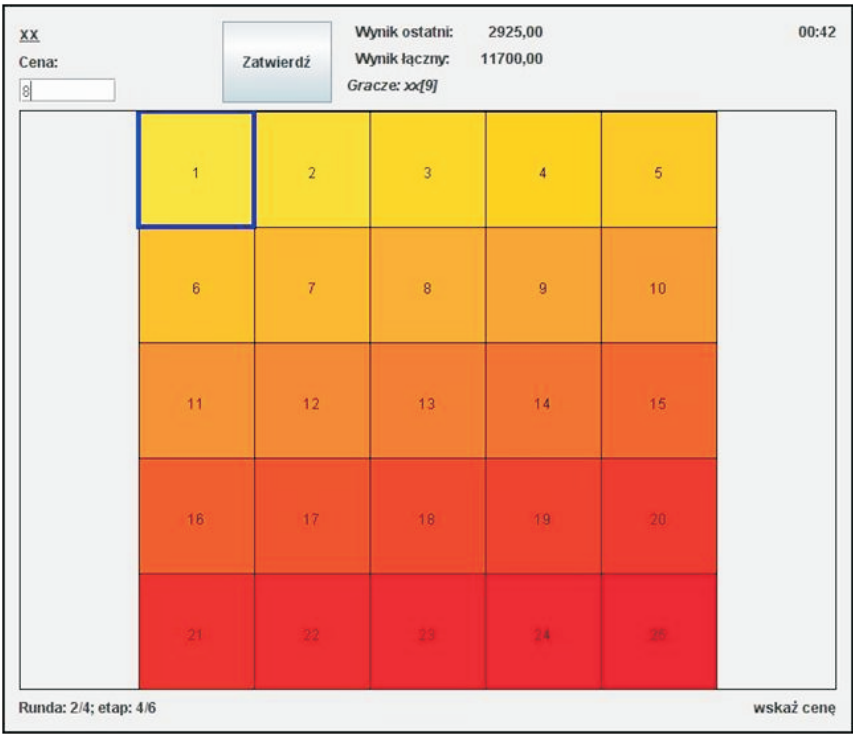
Java jest dostarczana razem z bardzo bogatym zestawem bibliotek standardowych. Oznacza to, że program napisany w Javie gwarantuje, że będzie w swojej pracy mógł skorzystać z określonych zestawów funkcji. W ten sposób praca nad programem zostaje skrócona, a same programy po skompilowaniu zajmują mniej miejsca, potrzebne biblioteki są częścią maszyny wirtualnej.

Java ma bardzo starannie zaprojektowaną składnię. Od lat składnia ta jest stabilna, wersje Javy są kompatybilne w dół, co oznacza, że stare programy będzie można kompilować za pomocą nowych wersji kompilatora. Składnia Javy jest czytelna, co więcej opis programu można zawrzeć w kodzie źródłowym, a następnie wytworzyć z niego dokumentację. Przyzwoita dokumentacja jest bardzo ważnym elementem dobrego oprogramowania. Włączenie dokumentacji bezpośrednio w kod źródłowy bardzo ułatwia pracę nad programem.

Java zyskała dużą popularność, dzięki czemu pojawiło się wiele narzędzi do pracy z tym językiem. Dostępne są niezależne kompilatory i wiele środowisk do tworzenia programów w Javie. Większość z nich to środowiska graficzne. Bardzo łatwo znaleźć w sieci dokumentację i przykłady. Większość narzędzi programistycznych jest darmowa.

Pierwsza wersja oprogramowania do tworzenia eksperymentów powstała na Wydziale Nauk Ekonomicznych Uniwersytetu Warszawskiego jako część pracy magisterskiej Pawła Kowala. Stworzony wtedy silnik pozwalał na tworzenie połączeń sieciowych między uczestnikami eksperymentu a prowadzącym. Niestety tworzenie eksperymentów na podstawie tego silnika wymagało znacznych umiejętności programistycznych. Jednak wyniki prac były

bardzo obiecujące. Oprogramowana przez Michała Ejdysa na podstawie tego silnika platforma do tworzenia eksperymentów przestrzennych miała możliwości graficzne o niespotykanych w innych programach możliwościach (por. rys. 1.). LabSEE (*Laboratory of Social and Economic Experiments*) w obecnym kształcie jest dziełem Roberta Borowskiego, który, korzystając jedynie z doświadczeń pierwszej wersji, napisał na nowo silnik i interfejs użytkownika, znacznie zbliżając obecną wersję do postulatów efektywnego tworzenia eksperymentów on-line. LabSEE jest połączeniem uniwersalności środowiska Java oraz przyjaznego interfejsu użytkownika znanego z programu z-Tree.



Rysunek 1.

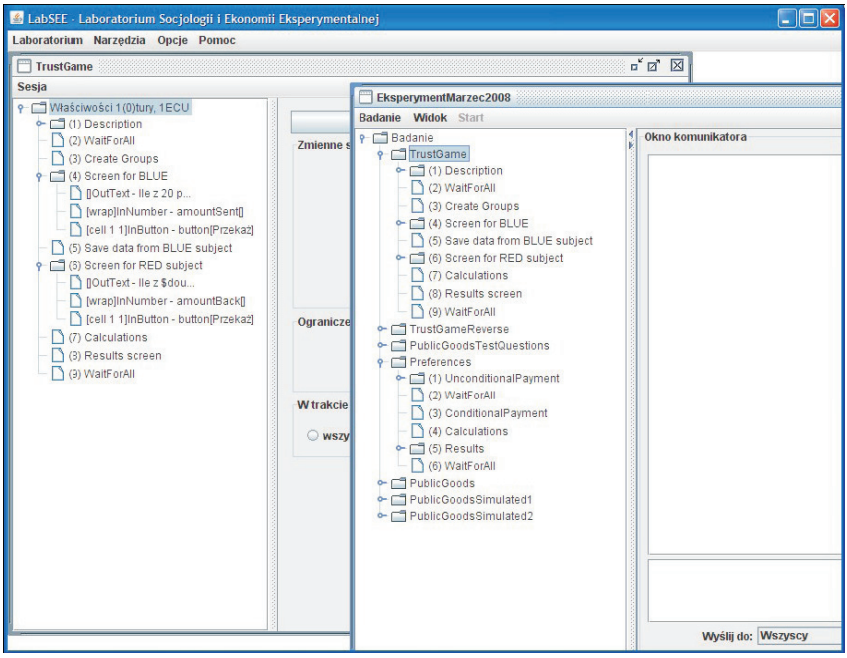
Możliwości graficzne w grach przestrzennych
Źródło: [Kopczewski, Kusztelak, Pogorzelski, 2008].

LabSEE jest oprogramowaniem typu klient-serwer. Do uruchomienia potrzebuje zainstalowanej na komputerze wirtualnej maszyny Javy minimum w wersji 6. poprawka 3. Aby móc korzystać z aplikacji klienta wystarczy środowisko uruchomieniowe JRE, natomiast — by móc pracować z aplikacja serwera — potrzebne jest środowisko programistyczne JDK[5]. W wyznaczonym

[5] Najnowsze środowiska Javy można pobrać ze strony producenta: http://java.sun.com/javase/downloads/.

czasie uruchomiony zostaje serwer, a na komputerach graczy programy klienckie. Komputery łączą się w sieć i możliwa jest gra on-line. Przed uruchomieniem programu klienckiego należy sprawdzić wersję i stan instalacji Javy. Aby połączyć się z aplikacją serwera potrzebne jest przekazanie do aplikacji klienta informacji o adresie IP i porcie, na którym nasłuchuje serwer. Ponadto można zmienić domyślny język na inny, przekazując do aplikacji odpowiednia lokalizację.

LabSEE narzucą pewną strukturę badania eksperymentalnego, które ma być realizowane. Wprowadzony jest podział badania na sesje stanowiące logiczną całość, która może być powtarzana, oraz nie ma z reguły przyczynowo-skutkowego związku z innymi sesjami. O podziale badania na sesje decyduje użytkownik. Na przykład przeprowadzenie dylematu więźnia powtarzanego 10 razy z tymi samymi parametrami tabeli wypłat może odpowiadać pojedynczej sesji. Przeprowadzenie na tych samych uczestnikach następnego eksperymentu z zakresu dóbr publicznych odpowiadać może kolejnej sesji. Kolejność sesji w badaniu może być ustalana i reorganizowana dowolnie. Zalecane jest stworzenie najpierw oddzielnie każdej sesji, a następnie ich połączenie w ramach całego badania. Po połączeniu sesji w badanie istnieje nadal możliwość edycji poszczególnych sesji, zarówno przed rozpoczęciem badania, jak i w trakcie jego trwania. Strukturę przykładowego badania oraz budowę jednej z jego sesji przedstawia poniższy rysunek.



Rysunek 2.

Struktura badania oraz pojedynczej sesji

Sesja może być powtarzana wielokrotnie — każdy pełny przebieg sesji określany jest mianem tury, która może być próbna albo płatna. W turach próbnych nie są naliczane wygrane w postaci pieniężnej. W trakcie badania jego uczestnicy otrzymują punkty za swoje decyzje, które są automatycznie sumowane z tury na turę. Automatycznie też na koniec każdej tury jest wyliczana wielkość płatności, odpowiadająca zdobytej liczbie punktów w danej turze i sumowana do całkowitej płatności.

Do tworzenia programów wykorzystywany jest język programowania Java. Ekrany natomiast mogą być tworzone za pomocą interfejsu graficznego (prosta definicja ekranu) lub przy wykorzystanie języka Java. Ta druga metoda ma zastosowanie w przypadkach zaawansowanych, gdzie możliwości interfejsu graficznego nie pozwalają na stworzenie wymaganego ekranu. LabSEE pozwala na komunikację uczestników oraz prowadzącego w ramach przeprowadzanego badania. W tym celu wykorzystywany jest wbudowany komunikator. Okno komunikatora jest zawsze dostępne dla prowadzącego badanie. Możliwości komunikacji uczestników badania są definiowane odrębnie dla każdego ekranu oraz dodatkowo dla ekranu oczekiwania na dalszy przebieg badania dla danej sesji. Istnieje możliwość całkowitego wyłączenia komunikatora dla uczestników, zezwolenia na komunikację jedynie z prowadzącym lub własną grupą, albo pozostałymi grupami i uczestnikami.

Funkcjonalność LabSEE jest ograniczona jedynie możliwościami programistycznymi użytkownika. Ponieważ liczba programistów Java jest duża, więc koszt stworzenia dowolnego eksperymentu jest stosunkowo niski w porównaniu z programowaniem eksperymentów od podstaw. Niezależnie od tego obecnie tworzone są gotowe funkcje, które umożliwiają osobom o małych umiejętnościach programistycznych przygotowanie kilkunastu różnych gier i eksperymentów on-line.

Program oraz jego dokumentację można pobrać ze strony internetowej projektu[6]. Program udostępniany jest na licencji Creative Commons[7]. LabSEE jest częścią planowanego Laboratorium Nauk Społecznych on-line. W skład tego projektu wchodzi jeszcze serwis ankietowy, tworzony przez Roberta Borowskiego[8], oraz implementacja programu do obliczeń statystycznych on-line R-CRAN[9].

Bibliografia

Eckel Bruce, 2003, *Thinking in Java*, Wydanie 3, Edycja Polska, Helion, Gliwice.

Fischbacher Urs, 2007, *z-Tree: Zurich Toolbox for Ready-made Economic experiments*, „Experimental Economics", nr 10 (2), s. 171–178, http://www.iew.niuzh.ch/ztree/.

[6] www.ekonomiaeksperymentalna.edu.pl.

[7] Licencja Creative Commons — Uznanie autorstwa — Użycie niekomercyjne — Na tych samych warunkach 2.5.

[8] www.moje-ankiety.pl.

[9] www.r-ekonomia.pl.

Holt A. Charles, 2007, *Markets, Games and Strategic Behaviour: Recipes for Interactive Learning*, Pearson Addison Wesley, Boston, London.
Kagel J. H., Roth A. E., 1995, *Handbook of Experimental Economics*, Princeton University Press.
Kahneman D., Tversky A., 2000, *Choices, values and frames*, Cambridge University Press.
Kopczewski Tomasz, Kusztelak Przemysław, Pogorzelski Maciej, 2008, *Company size and spatial localization — game theory and experimental approach*, International Conference on Economic Science with Heterogeneous Interacting Agents, Warszawa.
Mill J. S., 1836, *On the definition of political economy and the method of investigation proper to it*, w: *Collected Works of John Stuart Mill*, Vol. 4, Toronto Press, 1967, s. 124.
Michałek T., 2007, *Badania eksperymentalne samoistnego powstawania nieuczciwości na rynku*, „Ekonomia", nr 19.
Stroustrup Bjarne, 2002, *Język C++*, WNT, Warszawa.

Strony internetowe

http://java.sun.com/
http://kuznets.fas.harvard.edu/~aroth/
http://msdn.microsoft.com/
http://research.cassel.ucla.edu/software.htm
http://webserver.econ1.uni-bonn.de/ratimage/
http://www.econlab.arizona.edu/
http://www.iew.niuzh.ch/ztree/
http://www.kirchkamp.de/
http://www.people.virginia.edu/cah2k/